Recurrent Networks, and Attention, for Statistical Machine Translation

Michael Collins, Columbia University
Mapping Sequences to Sequences

- Learn to map input sequences $x_1 \ldots x_n$ to output sequences $y_1 \ldots y_m$ where $y_m = \text{STOP}$.
- Can decompose this as

$$p(y_1 \ldots y_m|x_1 \ldots x_n) = \prod_{j=1}^{m} p(y_j|y_1 \ldots y_{j-1}, x_1 \ldots x_n)$$

- Encoder/decoder framework: use an LSTM to map $x_1 \ldots x_n$ to a vector $h^{(n)}$, then model

$$p(y_j|y_1 \ldots y_{j-1}, x_1 \ldots x_n) = p(y_j|y_1 \ldots y_{j-1}, h^{(n)})$$

using a “decoding” LSTM
The Computational Graph
Training A Recurrent Network for Translation

Inputs: A sequence of source language words \(x_1 \ldots x_n\) where each \(x_j \in \mathbb{R}^d\). A sequence of target language words \(y_1 \ldots y_m\) where \(y_m = \text{STOP}\).

Definitions: \(\theta^F\) = parameters of an “encoding” LSTM. \(\theta^D\) = parameters of a “decoding” LSTM. LSTM\((x^{(t)} , h^{(t-1)} ; \theta)\) maps an input \(x^{(t)}\) together with a hidden state \(h^{(t-1)}\) to a new hidden state \(h^{(t)}\). Here \(\theta\) are the parameters of the LSTM.
Training A Recurrent Network for Translation
(continued)

Computational Graph:

- Initialize $h^{(0)}$ to some values (e.g. vector of all zeros)
- **(Encoding step:)** For $t = 1 \ldots n$
  - $h^{(t)} = \text{LSTM}(x^{(t)}, h^{(t-1)}; \theta^F)$
- (Decoding step:) For $j = 1 \ldots m$
  - $\beta^{(j)} = \text{LSTM}\left(\text{CONCAT}(y_{j-1}, h^{(n)}), \beta^{(j-1)}; \theta^D\right)$
  - $l^{(j)} = V \times \text{CONCAT}(\beta^{(j)}, y_{j-1}, h^{(n)}) + \gamma$, $q^{(j)} = \text{LS}(l^{(j)})$
  - $o^{(j)} = -q^{(j)}y_j$
- **(Final loss is sum of losses:)**
  - $o = \sum_{j=1}^{m} o^{(j)}$
The Computational Graph
Greedy Decoding with A Recurrent Network for Translation

- **Encoding step:** Calculate $h^{(n)}$ from the input $x_1 \ldots x_n$
- **$j = 1$. Do:**
  - $y_j = \arg \max_y p(y|y_1 \ldots y_{j-1}, h^{(n)})$
  - $j = j + 1$
  - **Until:** $y_{j-1} = \text{STOP}$
Greedy Decoding with A Recurrent Network for Translation

Computational Graph:

- Initialize $h^{(0)}$ to some values (e.g. vector of all zeros)
- (Encoding step:) For $t = 1 \ldots n$
  - $h^{(t)} = \text{LSTM}(x^{(t)}, h^{(t-1)}; \theta^F)$
- Initialize $\beta^{(0)}$ to some values (e.g., vector of all zeros)
- (Decoding step:) $j = 1$. Do:
  - $\beta^{(j)} = \text{LSTM(CONCAT}(y_{j-1}, h^{(n)}), \beta^{(j-1)}; \theta^D)$
  - $l^{(j)} = V \times \text{CONCAT}(\beta^{(j)}, y_{j-1}, h^{(n)}) + \gamma$
  - $y_j = \arg\max_y l^{(j)}$
  - $j = j + 1$
- Until $y_{j-1} = \text{STOP}$
- Return $y_1 \ldots y_{j-1}$
A bi-directional LSTM (bi-LSTM) for Encoding

**Inputs:** A sequence $x_1 \ldots x_n$ where each $x_j \in \mathbb{R}^d$.

**Definitions:** $\theta^F$ and $\theta^B$ are parameters of a forward and backward LSTM.

**Computational Graph:**

- $h^{(0)}, \eta^{(n+1)}$ are set to some initial values.
- For $t = 1 \ldots n$
  - $h^{(t)} = \text{LSTM}(x^{(t)}, h^{(t-1)}; \theta^F)$
- For $t = n \ldots 1$
  - $\eta^{(t)} = \text{LSTM}(x^{(t)}, \eta^{(t+1)}; \theta^B)$
- For $t = 1 \ldots n$
  - $u^{(t)} = \text{CONCAT}(h^{(t)}, \eta^{(t)}) \Leftarrow \text{encoding for position } t$
The Computational Graph
Incorporating *Attention*

- Old decoder:
  - $c^{(j)} = h^{(n)} \Leftarrow \text{context used in decoding at } j\text{'th step}$
  - $\beta^{(j)} = \text{LSTM} (\text{CONCAT}(y^{j-1}, c^{(j)}), \beta^{(j-1)}; \theta^D)$
  - $l^{(j)} = V \times \text{CONCAT}(\beta^{(j)}, y^{j-1}, c^{(j)}) + \gamma$
  - $y_j = \arg \max_y l_y^{(j)}$
Incorporating \textit{Attention}

- New decoder:
  - Define
    \[
    c(j) = \sum_{i=1}^{n} a_{i,j} u^{(i)}
    \]
    where
    \[
    a_{i,j} = \frac{\exp\{s_{i,j}\}}{\sum_{i=1}^{n} s_{i,j}}
    \]
    and
    \[
    s_{i,j} = A(\beta^{(j-1)}, u^{(i)}; \theta^A)
    \]
    where \(A(\ldots)\) is a non-linear function (e.g., a feedforward network) with parameters \(\theta^A\)
Greedy Decoding with Attention

- (Decoding step:) $j = 1$. Do:
  - For $i = 1 \ldots n$, 
    $$s_{i,j} = A(\beta^{(j-1)}, u^{(i)}; \theta^A)$$
  - For $i = 1 \ldots n$, 
    $$a_{i,j} = \frac{\exp\{s_{i,j}\}}{\sum_{i=1}^n s_{i,j}}$$
  - Set $c^{(j)} = \sum_{i=1}^n a_{i,j} u^{(i)}$
  - $\beta^{(j)} = \text{LSTM} (\text{ CONCAT}(y_{j-1}, c^{(j)}), \beta^{(j-1)}; \theta^D)$
  - $l^{(j)} = V \times \text{ CONCAT}(\beta^{(j)}, y_{j-1}, c^{(j)}) + \gamma$
  - $y_j = \arg \max_y l^{(j)}$
  - $j = j + 1$
  - Until $y_{j-1} = \text{STOP}$
  - Return $y_1 \ldots y_{j-1}$
Training with Attention

▶ (Decoding step:) For $j = 1 \ldots m$
  ▶ For $i = 1 \ldots n$, 
    \[ s_{i,j} = A(\beta^{(j-1)}, u^{(i)}; \theta^A) \]
  ▶ For $i = 1 \ldots n$, 
    \[ a_{i,j} = \exp\{s_{i,j}\} \frac{\sum_{i=1}^{n} s_{i,j}}{\sum_{i=1}^{n} s_{i,j}} \]
  ▶ Set $c^{(j)} = \sum_{i=1}^{n} a_{i,j} u^{(i)}$
  ▶ $\beta^{(j)} = \text{LSTM}([y_{j-1}, c^{(j)}], \beta^{(j-1)}; \theta^D)$
  ▶ $l^{(j)} = V \times \text{CONCAT}(\beta^{(j)}, y_{j-1}, c^{(j)}) + \gamma$, $q^{(j)} = \text{LS}(l^{(j)})$
  \[ o^{(j)} = -q_{y_j}^{(j)} \]
  ▶ (Final loss is sum of losses:)

\[ o = \sum_{j=1}^{m} o^{(j)} \]
The Computational Graph
### Table 10: Mean of side-by-side scores on production data

<table>
<thead>
<tr>
<th>Direction</th>
<th>PBMT</th>
<th>GNMT</th>
<th>Human</th>
<th>Relative Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>English → Spanish</td>
<td>4.885</td>
<td>5.428</td>
<td>5.504</td>
<td>87%</td>
</tr>
<tr>
<td>English → French</td>
<td>4.932</td>
<td>5.295</td>
<td>5.496</td>
<td>64%</td>
</tr>
<tr>
<td>English → Chinese</td>
<td>4.035</td>
<td>4.594</td>
<td>4.987</td>
<td>58%</td>
</tr>
<tr>
<td>Spanish → English</td>
<td>4.872</td>
<td>5.187</td>
<td>5.372</td>
<td>63%</td>
</tr>
<tr>
<td>French → English</td>
<td>5.046</td>
<td>5.343</td>
<td>5.404</td>
<td>83%</td>
</tr>
<tr>
<td>Chinese → English</td>
<td>3.694</td>
<td>4.263</td>
<td>4.636</td>
<td>60%</td>
</tr>
</tbody>
</table>

From *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*, Wu et al. 2016. Human evaluations are on a 1-6 scale (6 is best). PBMT is a phrase-based translation system, using IBM alignment models as a starting point.
Figure 6: Histogram of side-by-side scores on 500 sampled sentences from Wikipedia and news websites for a typical language pair, here English → Spanish (PBMT blue, GNMT red, Human orange). It can be seen that there is a wide distribution in scores, even for the human translation when rated by other humans, which shows how ambiguous the task is. It is clear that GNMT is much more accurate than PBMT.
Conclusions

- Directly model

\[
p(y_1 \ldots y_m | x_1 \ldots x_n) = \prod_{j=1}^{m} p(y_j | y_1 \ldots y_{j-1}, x_1 \ldots x_n)
\]

- Encoding step: map \( x_1 \ldots x_n \) to \( u^{(1)} \ldots u^{(n)} \) using a bidirectional LSTM

- Decoding step: use an LSTM in decoding together with attention