Natural Language Processing

Neural semantic parsing
Plan

• Sequence to sequence models
• LSTMs and GRUs
• Attention
• Pointer networks
• Weak supervision
Sequence to sequence
Semantic parsing

• We saw methods for translating natural language to logical form by constructing trees

• This works both when we have logical forms as well as denotations for supervision

• If we have logical forms as supervision, we can use an extremely popular neural architecture called sequence to sequence
Sequence to sequence

How tall is Lebron James?
HeightOf.LebronJames

[How, tall, is, Lebron, James, ?]
[HeightOf, ., LebronJames]

What do we gain? What do we lose?
High level

• We will build a complex, heavily-parameterized but differentiable function that maps a natural language statement $x$ to a logical form $y$ (or translation, or summary, or answer to a question…)

• We will define a loss (often cross entropy) that tells us how good is our prediction w.r.t the ground truth

• We will search for parameters that minimize the loss over the training set with SGD.

• We will compute gradients with auto-differentiation packages
Applications

• Machine translation
• Semantic parsing
• Question answering
• Summarization
• Dialogue
• …
Sequence to sequence

[How, tall, is, Lebron, James, ?]

[HeightOf, ., LebronJames]
Recurrent neural networks

Input: \( w_1, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_T \), \( w_i \in \mathbb{R}^\nu \)

Model: \( x_t = W^{(e)} \cdot w_t \), \( W^{(e)} \in \mathbb{R}^{d \times \nu} \)

\[
    h_t = \sigma (W^{(hh)} \cdot h_{t-1} + W^{(hx)} \cdot x_t), \quad W^{(hh)} \in \mathbb{R}^{D_h \times D_h}, \quad W^{(hx)} \in \mathbb{R}^{D_h \times d}
\]

\[
    \hat{y}_t = \text{softmax}(W^{(s)} \cdot h_t), \quad W^{(s)} \in \mathbb{R}^{\nu \times D_h}
\]
Encoder

An RNN without the output layer

How  tall  is  Lebron  James  ?  Vector
An RNN without the input layer

HeightOf (0.7)  (0.99)
WeightOf (0.2)  LebronJames(0.001)
Seq2seq (v1.0)

Encoder:
\[ h_0^e = 0, \ h_t^e = \text{RNN}_e(h_{t-1}^e, x_t) \]

Decoder:
\[ h_0^d = \text{RNN}_d(h_{|x|}^e), \ h_t^d = \text{RNN}_d(h_{t-1}^d) \]
\[ y_t = \text{softmax}(W^{(s)} h_t^d) \]

Model:
\[ p(y \mid x) = \prod_t p(y_t \mid y_1, \ldots, y_{t-1}, x) = \prod_t p(y_t \mid h_t^d) \]

Training is finding parameters that minimize cross entropy over tokens:
\[ \sum_i \log p_\theta(y^{(i)} \mid x^{(i)}) \]
Seq2seq (v1.0)

- Training is done with SGD on top of standard auto-diff packages
- At training time decoding is done as many steps as the training example (with a stopping symbol)
- At test time we output the argmax token of every time step and stop when we output the stopping symbol.
Seq2seq (v2.0)
Seq2seq (v2.0)

Encoder:
\[ h_0^e = 0, h_t^e = \text{RNN}_e(h_{t-1}^e, x_t) \]

Decoder:
\[ h_t^d = \text{RNN}_d(h_{t-1}^d, h_{|x|}^e, y_{t-1}) \]
\[ y_t = \text{softmax}(W^{(s)}h_t^d) \]

Model:
\[ p(y \mid x) = \prod_t p(y_t \mid y_1, \ldots, y_{t-1}, x) = \prod_t p(y_t \mid h_t^d) \]

Training is finding parameters that minimize cross entropy over tokens:
\[ \sum_i \log p_\theta(y^{(i)} \mid x^{(i)}) \]
Bidirectional encoder

How tall is Lebron James?
Bidirectional encoder

Encoder:

\[ h_f^t = 0, \quad h_f^t = \text{RNN}_f(h_{t-1}, x_t) \]
\[ h_b^t = 0, \quad h_b^t = \text{RNN}_b(h_{t+1}, x_t) \]

Decoder:

\[ h_d^t = \text{RNN}_d(h_d^{t-1}, h_f^t, h_b^t, y_{t-1}) \]

An extremely successful model (state-of-the-art), when using more sophisticated cells (LSTMs, GRUs).
Stacked RNNs

- For encoder or decoder

How tall is Lebron James?
Stacked RNN

- For stacked RNN, we need to have an output to each state, usually the hidden state itself is used as the input to the next layer.
- Empirically stacking RNNs is often better than just increasing the dimensionality.
- For example, Google’s NMT system uses 8 layers at both encoding and decoding time.
Efficiency

- RNNs are not very efficient in terms of parallelization.
- You cannot compute $h_t$ before computing $h_{t-1}$ (compared to bag of words or convolutional neural networks).
- This becomes a problem for tasks where one needs to read long documents (summarization).
Beam search

- At test time, decoding is greedy - we output the symbol that has highest probability. **Not** guaranteed to produce the highest probability sequence.

- Improved substantially with a small beam.

- At decoding step \( t \), we consider \( K \) most probable sequence prefixes, and compute all possible continuations, score them, and keep the top-\( K \).

- Burden shift from search to learning again.

[Diagram with HeightOf (0.7) and WeightOf (0.2)]
Vanishing gradients (reminder)

\[ h_t = W^{(hh)} \sigma(h_{t-1}) + W^{(hx)} x_t \]

\[ \frac{\partial J_t(\theta)}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial J_t(\theta)}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_t}{\partial \theta} \]

\[ \frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} W^{(hh)^\top} \text{diag}(\sigma'(h_{i-1})) \]

\[ \left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq \left\| W^{(hh)^\top} \right\| \cdot \left\| \text{diag}(\sigma'(h_{i-1})) \right\| \leq \eta \]

The contribution of step k to the total gradient at step t is small
Solutions (slide from the past)

- Exploding gradient: gradient clipping
  - Re-normalize gradient to be less than C (~5)
  - Exploding gradients are easy to detect

- The problem is with the model!
  - Change it (LSTMs, GRUs)
  - Maybe later in this class
LSTMs and GRUs
LSTMs and GRUs

- **Bottom line**: use vector addition and not matrix-vector multiplication. Allows for better propagation of gradients to the past.
Gated Recurrent Unit (GRU)

- **Main insight**: add learnable gates to the recurrent unit that control the flow of information from the past to the present

- **Vanilla RNN**:
  \[
  h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)
  \]

- **Update and reset gates**:
  \[
  z_t = \sigma(W^{(z)}h_{t-1} + U^{(z)}x_t)
  \]
  \[
  r_t = \sigma(W^{(r)}h_{t-1} + U^{(r)}x_t)
  \]
Gated Recurrent Unit (GRU)

\[ z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}) \]
\[ r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1}) \]
\[ \tilde{h}_t = \tanh(W x_t + r_t \circ U h_{t-1}) \]
\[ h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \]

- Use the gates to control information flow
  - If \( z=1 \), we simply copy the past and ignore the present (note that gradient will be 1)
  - If \( z=0 \), then we have an RNN like update, but we are also free to reset some of the past units, and if \( r=0 \), then we have no memory of the past
  - The + in the last equation is crucial
Illustration
Long short term memories (LSTMs)

- $z$ has been split into $i$ and $f$
- There is no $r$
- There is a new gate $o$ that distinguishes between the memory and the output.
- $c$ is like $h$ in GRUs
- $h$ is the output

\[
\begin{align*}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \\
\tilde{c}_t &= \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \tanh(c_t)
\end{align*}
\]
Illustration
Illustration

Chris Olah’s blog
More GRU intuition from Stanford

• Go over sequence of slides from Chris Manning
Machine translation

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahdanau et al. [2]</td>
<td>28.45</td>
</tr>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Single forward LSTM, beam size 12</td>
<td>26.17</td>
</tr>
<tr>
<td>Single reversed LSTM, beam size 12</td>
<td>30.59</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 1</td>
<td>33.00</td>
</tr>
<tr>
<td>Ensemble of 2 reversed LSTMs, beam size 12</td>
<td>33.27</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 2</td>
<td>34.50</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 12</td>
<td><strong>34.81</strong></td>
</tr>
</tbody>
</table>

Table 1: The performance of the LSTM on WMT’14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Cho et al. [5]</td>
<td>34.54</td>
</tr>
<tr>
<td>Best WMT’14 result [9]</td>
<td><strong>37.0</strong></td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single forward LSTM</td>
<td>35.61</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single reversed LSTM</td>
<td>35.85</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs</td>
<td><strong>36.5</strong></td>
</tr>
<tr>
<td>Oracle Rescoring of the Baseline 1000-best lists</td>
<td>~45</td>
</tr>
</tbody>
</table>

Sequence to Sequence Learning by Sutskever et al. 2014
Advantages of seq2seq

• Simplicity

• Distributed representations of words and phrases

• Better use of context (history) at encoding and decoding time

• Neural networks seem to be very good at generating text (for MT, summarization, QA, etc.)
Summary

• Sequence to sequence models map a sequence of symbols to another sequence of symbols - very common in NLP!

• LSTMs and GRUs allow this to work for long sequences (~30-100 steps)

• Results in state-of-the-art performance in many cases

• But not always! Attention!
Attention
The problem

What if the source is very long? “

how tall is the NBA player that has won the most NBA titles before he reached the age of 28?”

This is fixed size!

[How, tall, is, ...]

Vector

Decoder

[HeightOf, ., ...]
Attention

Treat source representations as memory

How  tall  is  Lebron  James  ?

Decide what to read from memory when decoding
Alignment

• What are the important words when we decide on the next symbol at decoding time?

• Alignments are heavily used in traditional MT

[HeightOf, ., LebronJames]

[How, tall, is, Lebron, James, ?]

• We will learn to perform the alignment as we decode
Learning alignment in MT

Decoding

Replace a fixed vector with a time-variable vector

Intuition: before generating a word we softly align to the relevant words in the source!
Attention

To compute $c_t$:  

$\forall i, s_i = \text{score}(h^{t-1}_d, h^i_e)$  

$\alpha = \text{softmax}(s)$

soft alignment!
Attention

To compute $c_t$:

$$c_t = \sum_i \alpha_i h^i_e$$

Diagram:

- tall
- Lebron
- James
- ?
- $<s>$
- HeightOf

Diagram shows a series of nodes connected by arrows, indicating the flow of information or computation.
Attention

To compute $c_t$:

$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$

$$\alpha = \text{softmax}(s)$$

soft alignment!
Attention example

(b) what’s first class fare round trip from ci0 to ci1
(c) what is the earliest flight from ci0 to ci1 tomorrow
Attention scoring function

- Options for scoring function:
  - Dot-product
  - Bilinear map
  - Single layer neural net
  - Multiple layer neural net

\[
\begin{align*}
\text{score}(h_{d}^{t-1}, h_{e}^{i}) &= h_{d}^{t-1\top} h_{e}^{i} \\
\text{score}(h_{d}^{t-1}, h_{e}^{i}) &= h_{d}^{t-1\top} W h_{e}^{i} \\
\text{score}(h_{d}^{t-1}, h_{e}^{i}) &= v^{\top} \tanh(W_{1} h_{d}^{t-1} + W_{2} h_{e}^{i})
\end{align*}
\]
Semantic parsing

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCISSOR (Ge and Mooney, 2005)</td>
<td>72.3</td>
</tr>
<tr>
<td>KRISP (Kate and Mooney, 2006)</td>
<td>71.7</td>
</tr>
<tr>
<td>WASP (Wong and Mooney, 2006)</td>
<td>74.8</td>
</tr>
<tr>
<td>λ-WASP (Wong and Mooney, 2007)</td>
<td>86.6</td>
</tr>
<tr>
<td>LSTM (Lu et al., 2008)</td>
<td>81.8</td>
</tr>
<tr>
<td>ZCO5 (Zettlemoyer and Collins, 2005)</td>
<td>79.3</td>
</tr>
<tr>
<td>ZCO7 (Zettlemoyer and Collins, 2007)</td>
<td>86.1</td>
</tr>
<tr>
<td>UBL (Kwiatkowski et al., 2010)</td>
<td>87.9</td>
</tr>
<tr>
<td>FUBL (Kwiatkowski et al., 2011)</td>
<td>88.6</td>
</tr>
<tr>
<td>KCAzi3 (Kwiatkowski et al., 2013)</td>
<td>89.0</td>
</tr>
<tr>
<td>DC5+1 (Liang et al., 2013)</td>
<td>87.9</td>
</tr>
<tr>
<td>TISP (Zhao and Huang, 2015)</td>
<td>88.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>t-Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</td>
<td>28.9</td>
<td>20.2</td>
<td>41.7</td>
</tr>
<tr>
<td>phrasal</td>
<td>19.3</td>
<td>11.3</td>
<td>35.3</td>
</tr>
<tr>
<td>sync</td>
<td>18.1</td>
<td>10.6</td>
<td>35.1</td>
</tr>
<tr>
<td>classifier</td>
<td>48.5</td>
<td>35.2</td>
<td>48.4</td>
</tr>
<tr>
<td>posclass</td>
<td>50.0</td>
<td>36.9</td>
<td>49.3</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>54.3</td>
<td>39.2</td>
<td>50.1</td>
</tr>
<tr>
<td>− attention</td>
<td>54.0</td>
<td>37.9</td>
<td>49.8</td>
</tr>
<tr>
<td>− argument</td>
<td>53.9</td>
<td>38.6</td>
<td>49.7</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>55.2</td>
<td>40.1</td>
<td>50.4</td>
</tr>
<tr>
<td>− attention</td>
<td>54.3</td>
<td>38.2</td>
<td>50.0</td>
</tr>
</tbody>
</table>

(a) Omit non-English.

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>t-Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</td>
<td>36.8</td>
<td>25.4</td>
<td>49.0</td>
</tr>
<tr>
<td>phrasal</td>
<td>27.8</td>
<td>16.4</td>
<td>39.9</td>
</tr>
<tr>
<td>sync</td>
<td>26.7</td>
<td>15.5</td>
<td>37.6</td>
</tr>
<tr>
<td>classifier</td>
<td>64.8</td>
<td>42.2</td>
<td>56.5</td>
</tr>
<tr>
<td>posclass</td>
<td>67.2</td>
<td>50.4</td>
<td>57.7</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>68.8</td>
<td>50.5</td>
<td>60.3</td>
</tr>
<tr>
<td>− attention</td>
<td>68.7</td>
<td>48.9</td>
<td>59.5</td>
</tr>
<tr>
<td>− argument</td>
<td>68.8</td>
<td>50.4</td>
<td>59.7</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>69.6</td>
<td>51.4</td>
<td>60.4</td>
</tr>
<tr>
<td>− attention</td>
<td>68.7</td>
<td>49.5</td>
<td>60.2</td>
</tr>
</tbody>
</table>

(b) Omit non-English & unintelligible.

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>t-Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</td>
<td>43.3</td>
<td>32.3</td>
<td>56.2</td>
</tr>
<tr>
<td>phrasal</td>
<td>37.2</td>
<td>23.5</td>
<td>45.5</td>
</tr>
<tr>
<td>sync</td>
<td>36.5</td>
<td>24.1</td>
<td>42.8</td>
</tr>
<tr>
<td>classifier</td>
<td>79.3</td>
<td>66.2</td>
<td>65.0</td>
</tr>
<tr>
<td>posclass</td>
<td>81.4</td>
<td>71.0</td>
<td>66.5</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>87.8</td>
<td>75.2</td>
<td>73.7</td>
</tr>
<tr>
<td>− attention</td>
<td>88.3</td>
<td>73.8</td>
<td>72.9</td>
</tr>
<tr>
<td>− argument</td>
<td>86.8</td>
<td>74.9</td>
<td>70.8</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>89.7</td>
<td>78.4</td>
<td>74.2</td>
</tr>
<tr>
<td>− attention</td>
<td>87.6</td>
<td>74.9</td>
<td>73.5</td>
</tr>
</tbody>
</table>

(c) > 3 Turkers agree with gold.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZCO5 (Zettlemoyer and Collins, 2005)</td>
<td>84.6</td>
</tr>
<tr>
<td>UBL (Kwiatkowski et al., 2010)</td>
<td>71.4</td>
</tr>
<tr>
<td>FUBL (Kwiatkowski et al., 2011)</td>
<td>82.8</td>
</tr>
<tr>
<td>GUSP-FULL (Poon, 2013)</td>
<td>74.8</td>
</tr>
<tr>
<td>GUSP++ (Poon, 2013)</td>
<td>83.5</td>
</tr>
<tr>
<td>TISP (Zhao and Huang, 2015)</td>
<td>84.2</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>84.2</td>
</tr>
<tr>
<td>− attention</td>
<td>72.3</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>84.6</td>
</tr>
</tbody>
</table>

Table 3: Evaluation results on GCO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZCO5 is used for all other systems.

Table 4: Evaluation results on ATIS.

Table 5: Evaluation results on TPTT.
Machine translation

![Graph showing BLEU scores with and without attention]

- Ours, no attn (BLEU 13.9)
- Ours, local-p attn (BLEU 20.9)
- Ours, best system (BLEU 23.0)
- WMT'14 best (BLEU 20.7)
- Jeans et al., 2015 (BLEU 21.6)

Sent Lengths vs. BLEU scores graph.
Coverage

• Caption generation

• How do we make sure we cover the source?
  • Penalize for source patches/words that are not aligned to any target word

\[
\sum_{\text{patch}} \left( 1 - \sum_{\text{word}} \alpha_{\text{patch,word}} \right)^2
\]
Attention variants

- Feed attention vectors as input at decoding time to try to learn coverage
- Add some term for preferring alignments that are monotonic
- Prefer limited fertility
- Use it for aligning pairs of text like (q,a) or paraphrase pairs
Summary

- Attention has enabled getting state-of-the-art performance for transduction scenarios
- Allows to softly align each token in a sequence of text to another sequence
Pointer networks
Problem

- Often at test time you need to translate entities you have never seen
- If we define the target vocabulary with the training set, we will never get it right
- In addition, translation for those entities is often simply copying

How tall is Dreymond Green?
HeightOf.DreymondGreen
Solution 1

- Mask entities
- Translate
- Bring back entities
  - But if there are many entities
  - How do you identify entities?

```
How tall is <e>?
HeightOf.<e>
```
Idea

• When we translate a sentence, the probability of a word increases once we see it.
  
  • \( P(\text{“pokemen”}) \) is low
  
  • \( P(\text{“pokemon”} \mid \text{“the pokemon company”}) \) is high
  
• Let’s allow outputting either words from a fixed target vocabulary or any word from the source sentence
Regular model

\[ p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \exp(U_w h_t) \]

HeightOf [3]
WeightOf [-1]
NumAssists [40]
( [9]
) [5.8]
. [3.7]
and [13]

Copying model

\[ p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \exp(U_w h_t) \]
\[ p(y_t = x_i \mid x, y_1, \ldots, y_{t-1}) \propto \exp(s_{ti}) \]

How tall is Dreymond Green?

Copying model

- Need to marginalize over the words since there could be repetitions
- At training this means that the true distribution is uniform over all correct tokens
- At test time we choose the highest probability token, but marginalize over the same instances of a token
Slight improvement

\[ p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \exp(U_w h_t) \]
\[ p(y_t = x_i \mid x, y_1, \ldots, y_{t-1}) \propto \exp(s_{ti}) \]

- These scores need to be calibrated
- We can just interpolate two distributions after normalization

\[ p_{\text{vocab}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(U h_t) \]
\[ p_{\text{copy}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(s_t) \]
\[ p(y_t \mid x, y_1, \ldots, y_{t-1}) = p_{\text{gen}} \cdot p_{\text{vocab}} + (1 - p_{\text{gen}}) \cdot p_{\text{copy}} \]
\[ p_{\text{gen}} = \sigma(w_1^\top h_t + w_2^\top c_t + w_3^\top y_{t-1}) \]
• Neural network for semantic parsing are based on sequence to sequence models

• These models are useful also for summarization, dialogue, question answering, paraphrasing, and other transduction tasks

• LSTMs and GRUs allowed to not forget to fast

• Attention added memory to circumvent the constant representation problem

• Pointer networks help in handling new words at test time

• Together you can often get models that are comparable to state-of-the-art without a grammar
Weak supervision
Weak supervision

• We have assumed that we have as input pairs of natural language and logical form

• In practice those are hard to collect and we usually have (language, denotation) pairs

<table>
<thead>
<tr>
<th>Heavy supervision</th>
<th>Light supervision</th>
</tr>
</thead>
<tbody>
<tr>
<td>How tall is Lebron James?</td>
<td>How tall is Lebron James?</td>
</tr>
<tr>
<td>HeightOf.LebronJames</td>
<td>203cm</td>
</tr>
<tr>
<td>What is Steph Curry’s daughter called?</td>
<td>What is Steph Curry’s daughter called?</td>
</tr>
<tr>
<td>ChildrenOf.StephCurry ⊎ Gender.Female</td>
<td>Riley Curry</td>
</tr>
<tr>
<td>Youngest player of the Cavaliers</td>
<td>Youngest player of the Cavaliers</td>
</tr>
<tr>
<td>arg min(PLAYEROF.Cavaliers, BirthDateOf)</td>
<td>Kyrie Irving</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The problem

• In sequence to sequence we trained end-to-end with SGD, minimizing the cross entropy loss of every token

• Here we don’t have tokens

• **Suggestion**: generate the program token by token, execute, and minimize cross entropy over denotations

• **Problem**: The loss is not a differentiable function of the input because we don’t input gold tokens
Can we do something similar with a seq2seq model?
Markov Decision Process

- Sequence of states, actions and rewards
  - $s_0, s_1, s_2, \ldots, s_T$ from a set $S$
  - $a_0, a_1, a_2, \ldots, a_T$ from a set $A$
    - Let’s assume a deterministic transition function $f:S\times A \rightarrow S$
  - $r_0, r_1, r_2, \ldots, r_T$ given by a reward function $r(s,a)$
- We want a policy $\pi(a \mid s)$ providing a distribution over actions that will maximize future reward
Seq2seq as MDP

- $s_t$: $h_t$
- $a_t$ is in $A(s_t)$
  - Either all symbols in the target vocabulary
  - All valid symbols if we check grammaticality
- $r_t$ is zero in all steps except the last. Then, it is 1 if execution results in a correct answer and 0 otherwise.

Liang et al, 2017, Guu et al., 2017
Seq2seq as MDP: policy

\[
p(z \mid x) = \prod_t p(z_t \mid x, z_0, \ldots, z_{t-1})
= \prod_t p(a_t \mid x, a_0, \ldots, a_{t-1})
= \prod_t \pi(a_t \mid s_t)
\]

\[
\pi(a_t \mid s_t) = \text{softmax}(W^{(s)}h_t)
\]

How do we learn?
Option 1: Maximum marginal likelihood

- Our data is language-dentation pairs \((x,y)\)
- We obtain \(y\) by constructing a logical form \(z\)
- We can use maximum marginal likelihood like before
- Interleave search and learning
  - Apply search to get candidate logical forms
  - Apply learning on these candidate
- Difference from before:
  - Search was done with CKY and learning was a globally-normalized model
  - Search can be done with beam search and we have a locally-normalized model
Maximum marginal likelihood

- $z$ is independent of $x$ conditioned on $y$

$$p_\theta(y \mid x) = \sum_z p_\theta(z \mid x) \cdot p(y \mid z)$$

$$= \sum_z p_\theta(z \mid x) R(z) = E_{p_\theta(z \mid x)}[R(z)]$$

$$\mathcal{L}_{\text{MML}}(\theta) = \log \prod_{(x,y)} p_\theta(y \mid x) = \log \prod_{(x,y)} E_{p_\theta(z \mid x)}[R(z)]$$

$$= \sum_{(x,y)} \log \sum_z p_\theta(z \mid x) \cdot R(z)$$
Gradient of MML

- Gradient has similar form to what we have seen in the past, except that we are not in a log-linear model. Let's assume a binary reward:

\[ \nabla_\theta \log \sum_z p_\theta(z \mid x) \cdot R(z) = \sum_z \frac{p_\theta(z) R(z) \nabla \log p_\theta(z \mid x)}{\sum'_{z'} p_\theta(z' \mid x) \cdot R(z')} \]

\[ = \sum_z p(z \mid x, R(z) = 1) \nabla \log p_\theta(z \mid x) \]

- Compute the gradient of the log probability for every logical form, and weight the gradient using the reward.
Computing the gradient

- We can not enumerate all of the logical forms.
- Instead we perform beam search as usual and get a beam $Z$ containing $K$ logical forms.
- We imagine that this beam is the entire set of possible logical forms:

$$
\sum_{z \in Z} p(z \mid x, R(z) = 1) \nabla \log p_\theta(z \mid x)
$$

- For every $z$ we can compute the gradient of $\log p(z \mid x)$ since this is now the usual seq2seq setup.
Option 2: policy gradient

- We would like to simply maximize our expected reward

\[
E_{p_\theta(z|x)}[R(z)] = \sum_{z} p_\theta(z \mid x) R(z)
\]

\[
\mathcal{L}_{RL}(\theta) = \sum_{(x,y)} \sum_{z} p_\theta(z \mid x) R(z) = \sum_{(x,y)} E_{p_\theta(z|x)}[R(z)]
\]

\[
\nabla \mathcal{L}_{RL}(\theta) = \sum_{(x,y)} \sum_{z} p_\theta(z \mid x) R(z) \nabla \log p_\theta(z \mid x)
\]

\[
= \sum_{(x,y)} E_{p_\theta(z|x)}[R(z) \nabla \log p_\theta(z \mid x)]
\]

- Weight the gradient by the product of the reward and the model probability
Computing the gradient

• Again, we can not sum over all logical forms

• But the gradient for every example is an expectation over a distribution we can sample from!

• So we can sample many logical forms, compute the gradient and sum them weighted by the product of the model probability and reward

• Again, for every sample this is regular seq2seq and we can compute an approximate gradient
Some differences

• Using MML with beam search is a biased estimator and has less exploration - we only observe the approximate top-K logical forms

• Using RL could be harder to train. If we have a correct logical form $z^*$ that has low probability at the beginning of training, then the contribution to the gradient would be very smaller and it would be hard to bootstrap.
Summary

• Training with a seq2seq model with weak supervision is problematic because the loss function is not a differentiable function of the input

• We saw both MML and RL approaches for getting around that

• In both we find a set of logical forms, compute the gradient for them like in supervised learning, and weight them in some way to form a final gradient

• This let’s us train with SGD

• It is still often hard to train - more next time