Natural Language Processing

Neural semantic parsing

Slides adapted from Richard Socher, Chris Manning, Ofir Press
Plan

• Sequence to sequence models
• Attention
• Pointer networks
• Weak supervision
Sequence to sequence
Conditional generation

<table>
<thead>
<tr>
<th>Input X</th>
<th>Output Y (Text)</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured Data</td>
<td>NL Description</td>
<td>NL Generation</td>
</tr>
<tr>
<td>English</td>
<td>Japanese</td>
<td>Translation</td>
</tr>
<tr>
<td>Document</td>
<td>Short Description</td>
<td>Summarization</td>
</tr>
<tr>
<td>Utterance</td>
<td>Response</td>
<td>Response Generation</td>
</tr>
<tr>
<td>Image</td>
<td>Text</td>
<td>Image Captioning</td>
</tr>
<tr>
<td>Speech</td>
<td>Transcript</td>
<td>Speech Recognition</td>
</tr>
</tbody>
</table>

Slide credit: Graham Neubig
How tall is Lebron James?
HeightOf.LebronJames

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

How tall is Lebron James?
¿Que tan alto es Lebron James?
High level

• We will build a 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: \( w_1, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_T \), \( w_i \in \mathbb{R}^\gamma \)

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

\[
\begin{align*}
  h_t &= \sigma(W^{(hh)} \cdot h_{t-1} + W^{(hx)} \cdot x_t), \quad W^{(hh)} \in \mathbb{R}^{D_h \times D_h}, \quad W^{(hx)} \in \mathbb{R}^{D_h \times d} \\
  \hat{y}_t &= \text{softmax}(W^{(s)} \cdot h_t), \quad W^{(s)} \in \mathbb{R}^{\gamma \times D_h}
\end{align*}
\]
Encoder

An RNN without the output layer

How → tall → is → Lebron → James → ? → Vector
Decoder (v1.0)

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:
\[ h_0^f = 0, \quad h_t^f = \text{RNN}_f(h_{t-1}, x_t) \]
\[ h_{|x|}^b = 0, \quad h_t^b = \text{RNN}_b(h_{t+1}, x_t) \]

Decoder:
\[ h_t^d = \text{RNN}_d(h_{t-1}^d, h_{|x|}^f, h_0^b, y_{t-1}) \]

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).

- Today we have the transformer architecture that can replace the RNN encoder.
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 \textit{top-K}.

• Burden shift from search to learning again.
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 (~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, ...]
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

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

Replace a fixed vector with a time-variable vector
Attention

To compute $c_t$: 

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

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

To compute $c_t$:

$$\forall i, s_i = \text{score}(h_{d}^{t-1}, h_{e}^{i})$$

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

To compute $c_t$:

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

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

```
4 → 2 → Lebron → James → ? → HeightOf
```
Attention

To compute $c_t$: 

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

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

$$\text{HeightOf}$$
Attention

To compute $c_t$: 

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

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

To compute \( c_t \):

\[
\forall i, s_i = \text{score}(h_{d-1}^t, h_e^i)
\]

\[
\alpha = \text{softmax}(s)
\]

\[
\begin{array}{cccc}
.644 & .087 & .237 & .032 \\
4 & 2 & 3 & 1
\end{array}
\]

\[
\begin{array}{cccc}
\text{tall} & \text{Lebron} & \text{James} & ? \\
\text{HeightOf} & \text{HeightOf}
\end{array}
\]
Attention

To compute $c_t$:

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

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

Soft alignment!
Attention

To compute $c_t$: $c_t = \sum_i \alpha_i h_e^i$
Attention

To compute $c_t$:

$$c_t = \sum_i \alpha_i h_e^i$$
Attention

To compute $c_t$: \[ \forall i, s_i = score(h_{d-1}^t, h_e^i) \]
\[ \alpha = \text{softmax}(s) \]

soft alignment!

.tall .476 .476 .024
 1 4 4 1

HeightOf

<\text{s}> HeightOf .
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

\[
\text{score}(h_{d}^{t-1}, h_{e}^{i}) = h_{d}^{t-1\top} h_{e}^{i}
\]
\[
\text{score}(h_{d}^{t-1}, h_{e}^{i}) = h_{d}^{t-1\top} W h_{e}^{i}
\]
\[
\text{score}(h_{d}^{t-1}, h_{e}^{i}) = v^{\top} \tanh(W_{1} h_{d}^{t-1} + W_{2} h_{e}^{i})
\]
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>-WASP (Wong and Mooney, 2007)</td>
<td>86.6</td>
</tr>
<tr>
<td>LNLZ08 (Lu et al., 2008)</td>
<td>81.8</td>
</tr>
<tr>
<td>ZC05 (Zettlemoyer and Collins, 2005)</td>
<td>72.3</td>
</tr>
<tr>
<td>ZC07 (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>KCAZ13 (Kwiatkowski et al., 2013)</td>
<td>89.0</td>
</tr>
<tr>
<td>DCS+L (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>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ2SEQ</td>
<td>84.6</td>
</tr>
<tr>
<td>- attention</td>
<td>72.9</td>
</tr>
<tr>
<td>- argument</td>
<td>68.6</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>87.1</td>
</tr>
<tr>
<td>- attention</td>
<td>76.8</td>
</tr>
</tbody>
</table>

Table 3: Evaluation results on GEO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZC05 is used for all other systems.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZC07 (Zettlemoyer and Collins, 2007)</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ2SEQ</td>
<td>84.2</td>
</tr>
<tr>
<td>- attention</td>
<td>75.7</td>
</tr>
<tr>
<td>- argument</td>
<td>72.3</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>84.6</td>
</tr>
<tr>
<td>- attention</td>
<td>77.5</td>
</tr>
</tbody>
</table>

Table 4: Evaluation results on ATIS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>+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>48.8</td>
<td>35.2</td>
<td>48.4</td>
</tr>
<tr>
<td>poseclass</td>
<td>50.0</td>
<td>36.9</td>
<td>49.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>+Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</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>+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>37.8</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>47.2</td>
<td>56.5</td>
</tr>
<tr>
<td>poseclass</td>
<td>67.2</td>
<td>50.4</td>
<td>57.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>+Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</td>
<td>68.3</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>+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>poseclass</td>
<td>81.4</td>
<td>71.0</td>
<td>66.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Channel</th>
<th>+Func</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</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>70.4</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 5: Evaluation results on IFTTT.
Machine translation

![Diagram showing BLEU scores for different translation models with and without attention. The graph plots BLEU scores against sentence lengths. The x-axis represents sentence lengths ranging from 10 to 70, while the y-axis represents BLEU scores ranging from 10 to 25. The graph includes lines and markers for different models, such as "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), and "Jeans et al., 2015" (BLEU 21.6).]
Coverage

• Caption generation

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

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

- Attention-based encoders **and decoders**

---

Slide credit: Lukasz Kaiser
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
Class 10
Projects

- Not everybody sent project descriptions
- You have until the end of the day
Next week

• Each team doing the research project will describe their project: motivation, plan, evaluation in 3-5 min.

• Please put a pdf with up to two slides in the shared folder.

• Not presenting will result in a 5-point deduction from the project grade. If you have good reason not to present, contact asap.
Attention

• Motivation: eliminate the bottleneck of a single vector in sequence to sequence models

• Framework:
  • Query vector $h_q$
  • Memory with key, value pairs $h_k, h_v$
  • $h_q$ is used to score each key $h_k$ and create a distribution over the memory
  • The distribution is used to compute a weighted average over the memory values $h_v$
Attention

• Seq2seq with LSTMs - one type of attention:
  
  • $h_q$: decoder hidden state, $h_k$: encoder hidden states, $h_v$: encoder hidden states
  
• Transformer has two additional attention types:
  
  • Encoder self-attention: $h_q = h_k = h_v$ are encoder hidden states
  
  • Decoder self-attention $h_q = h_k = h_v$ are encoder hidden states (when decoding token $t$, the decoder attends only to tokens $1…t-1$)
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

  • $P(\text{"pokemon" | "the pokemon company"})$ 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?

- HeightOf [3]
- WeightOf [-1]
- NumAssists [40]
  - ( [9]
  - ) [5.8]
  - . [3.7]
  - and [13]
- How [5]
- tall [1]
- is [-2]
- Dreymond [100]
- Green [100]
- ? [0]
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

\[ p_{\text{vocab}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(U h_t) \]
\[ p_{\text{copy}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(s_t) \]
\[ p(y_t \mid x, y_1, \ldots, y_{t-1}) = p_{\text{gen}} \cdot p_{\text{vocab}} + (1 - p_{\text{gen}}) \cdot p_{\text{copy}} \]
\[ p_{\text{gen}} = \sigma(w_1^\top h_t + w_2^\top c_t + w_3^\top y_{t-1}) \]
Illustration
Summary

• 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

• 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

### Heavy supervision

- How tall is Lebron James?
- HeightOf.LebronJames

### Light supervision

- What is Steph Curry’s daughter called?
- ChildrenOf.StephCurry \cap Gender.Female
- Youngest player of the Cavaliers
- arg min( PlayerOf.Cavaliers, BirthDateOf )

- How tall is Lebron James?
- 203cm
- What is Steph Curry’s daughter called?
- Riley Curry
- Youngest player of the Cavaliers
- Kyrie Irving
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
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

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

\[ = \prod_t p(a_t \mid x, a_0, \ldots, a_{t-1}) \]

\[ = \prod_t \pi(a_t \mid s_t) \]

\[ \pi(a_t \mid s_t) = \text{softmax}(W^{(s)} h_t) \]
Seq2seq as MDP: policy

\[ p(z | x) = \prod_t p(z_t | x, z_0, \ldots, z_{t-1}) \]

\[ = \prod_t p(a_t | x, a_0, \ldots, a_{t-1}) \]

\[ = \prod_t \pi(a_t | s_t) \]

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

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 by interleaving search and learning:
  - Search to get candidate logical forms
  - Learn from these candidates
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:

$$
\nabla_\theta \log \sum_z p_\theta(z \mid x) \cdot R(z) = \sum_z \frac{p_\theta(z \mid x)R(z) \nabla \log p_\theta(z \mid x)}{\sum'_{z'} p_\theta(z' \mid x) \cdot R(z')} \\
= \sum_z p(z \mid x, R(z) = 1) \nabla \log p_\theta(z \mid x)
$$

- 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