Neural Responder, Answerer, Enquirer, Copier, etc.

Lili Mou
doublepower.mou@gmail.com
Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning
Neural Responder

- seq2seq
- Attention

ID | Post | NRM-glo | NRM-loc | NRM-hyb | Rtr.-based
---|------|--------|--------|--------|----------
1  | 每年元旦都来高烧。 | 多休息，增强免疫力！ | 每年元旦都去医院看病 | 祝早日康复，身体健 | 高烧… |
    | High fever attacks me every New Year’s Day | Rest more to boost your immune system. | Go to see a doctor every New Year’s Day | Get well soon and stay healthy! | High fever… |
2  | 又老了一岁，感谢我们组，我很高兴 | 生日快乐！永远支持你！ | 生日快乐！永远年轻漂亮！ | 老了老了，岁月不饶人啊！ | 我只是老了一岁 |
    | I gain one more year. Grateful to my group, so happy | Happy birthday! Will stand by you forever! | Happy birthday! Wish you always stay young and pretty! | Getting old now. Time has no mercy. | I just gain only one more year. |
Neural Generative Question Answering

Jun Yin² Xin Jiang⁲ Zhengdong Lu²
Lifeng Shang⁲ Hang Li² Xiaoming Li¹
¹School of Electronic Engineering and Computer Science, Peking University
  {jun.yin, lxm}@pku.edu.cn
  ²Noah’s Ark Lab, Huawei Technologies
  {Jiang.Xin, Lu.Zhengdong, Shang.Lifeng, HangLi.HL}@huawei.com

“He is 2.29m and visible from space”

“How tall is Yao Ming?”
Neural QA: Interpreter

- Interpreter: Bi-RNN over the query

\[ \tilde{h}_t = [h_t; x_t] \]

“He is 2.29m and visible from space”

“How tall is Yao Ming?”
Neural QA: Enquirer

- Interpreter: $\tilde{h}_t = [h_t; x_t]$
- Enquirer: $S(Q, \tau) = \tilde{h}_Q^\top M u_\tau$

$\tilde{h}_Q$: Avg pooling of $h$

$u$: embedding of the tuple (sum of subject & predicate)

$M$: parameters

$e^{S(Q, \tau_k)}$  
$\sum_{k'} e^{S(Q, \tau_{k'})}$

“He is 2.29m and visible from space”

“How tall is Yao Ming?”

Normalized score (over 50 pre-selected candidates)
Neural QA: Answerer

- **Interpreter:** $\tilde{h}_t = [h_t; x_t]$
- **Enquirer:** $S(Q, \tau) = \tilde{h}_Q^T M u_\tau$
- **Answerer:** RNN generator w/ attent.

“He is 2.29m and visible from space”

$p(y_1, \ldots, y_{T_Q} | H_Q, r_Q; \theta) = p(y_1 | H_Q, r_Q; \theta) \prod_{t=2}^{T_Q} p(y_t | y_1, \ldots, y_{t-1}, H_Q, r_Q; \theta)$

Prob. that the word is generated from KB

$\frac{e^{S(Q, \tau_k)}}{\sum_{k'} e^{S(Q, \tau_{k'})}}$

Prob. that the word is generated by RNN w/ attent.

$p(y_t | s_t; \theta) = p(z_t = 0 | s_t; \theta) p(y_t | s_t, 0; \theta) + p(z_t = 1 | s_t; \theta) p(y_t | s_t, 1; \theta)$
### Table 3: Training and test accuracies

<table>
<thead>
<tr>
<th>Models</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval-based</td>
<td>40%</td>
<td>36%</td>
</tr>
<tr>
<td>NRM</td>
<td>15%</td>
<td>19%</td>
</tr>
<tr>
<td>GENQA</td>
<td>46%</td>
<td>47%</td>
</tr>
</tbody>
</table>

### Questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Answers generated by GENQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>三国演义谁写的？WHO wrote the Romance of the Three Kingdoms?</td>
<td>当然是罗贯中写的啊。Of course it is Luo Guanzhong who wrote it.</td>
</tr>
<tr>
<td>项羽是哪个国家的？Which country does Xiang Yu belong to?</td>
<td>楚国的皇帝。King of the Chu State.</td>
</tr>
<tr>
<td>还珠格格多少集？How many episodes does My Fair Princess have?</td>
<td>一共24集。24 episodes in total.</td>
</tr>
</tbody>
</table>
Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning
**Neural Enquirer:** Learning to Query Tables with Natural Language

Pengcheng Yin†* Zhengdong Lu† Hang Li† Ben Kao†
†Dept. of Computer Science
The University of Hong Kong
{pcyin, kao}@cs.hku.hk
‡Noah’s Ark Lab, Huawei Technologies
{Lu.Zhengdong, HangLi.HL}@huawei.com

**Disclaimer:** Details may vary.

**Logical Form:**

\[
\hat{F} = \text{where year < (select year, where host_city=Beijing)}, \text{argmax(host_city, #_duration)}
\]

**Query:** Which city hosted the longest Olympic game before the game in Beijing?

**Table:**

<table>
<thead>
<tr>
<th>year</th>
<th>host_city</th>
<th>#_duration</th>
<th>#_medals</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Sydney</td>
<td>20</td>
<td>2,000</td>
</tr>
<tr>
<td>2004</td>
<td>Athens</td>
<td>35</td>
<td>1,500</td>
</tr>
<tr>
<td>2008</td>
<td>Beijing</td>
<td>30</td>
<td>2,500</td>
</tr>
<tr>
<td>2012</td>
<td>London</td>
<td>40</td>
<td>2,300</td>
</tr>
</tbody>
</table>
Query Encoder

Which city hosted the longest Olympic game before the game in Beijing?

logical form $\tilde{F}$

where year<\(\text{select year, where host\_city=Beijing}\),
argmax(host\_city, #_duration)
Which city hosted the longest Olympic game before the game in Beijing?

logical form $\tilde{F}$

where $year$ = (select $year$, where $host\_city$ = Beijing),
argmax ($host\_city$, $\#\_duration$)
Query Encoder

- BiLSTM for query (Q) parsing
- Logical form (F) appears to be used for supervision only

\[
\begin{align*}
    h_t &= z_t h_{t-1} + (1 - z_t) \tilde{h}_t \\
    \tilde{h}_t &= \tanh(Wx_t + U(r_t \circ h_{t-1})) \\
    z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\
    r_t &= \sigma(W_r x_t + U_r h_{t-1})
\end{align*}
\]

Which city hosted the longest Olympic game before the game in Beijing?

Logical form \( \bar{F} \):

where year<\(\text{select}\) year, where host_city=Beijing),
argmax(host_city, #_duration)
Table Encoder

- Vector representation of each entry
  (a function of the entry and the field)

\[ e_{mn} = \text{DNN}_0([\mathbf{L}[w_{mn}]; f_n]) = \tanh(\mathbf{W} \cdot [\mathbf{L}[w_{mn}]; f_n] + b) \]
Executor

- **Reader**: To obtain a vector representation of a row

- **Annotator**: 

![Diagram showing the process of obtaining a vector representation of a row using DNNs and table annotations.]
Executor

- **Reader**: To obtain a vector representation of a row

- **Annotator**: To obtain yet another vector representation

\[
\text{Read Vector: } r_m^\ell = f_R(\mathcal{R}_m, \mathcal{F}_T, q, \mathcal{M}^{\ell-1}) \\
\text{Row Annotation: } a_m^\ell = f_A(r_m^\ell, q, \mathcal{M}^{\ell-1})
\]
Executor->Reader

Read Vector: \[ \mathbf{r}_m^\ell = f_R^\ell(\mathbf{R}_m, \mathcal{F}_T, \mathbf{q}, \mathcal{M}^{\ell-1}) \]

\[ \mathbf{r}_m^\ell = f_R^\ell(\mathbf{R}_m, \mathcal{F}_T, \mathbf{q}, \mathcal{M}^{\ell-1}) = \sum_{n=1}^{N} \tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}) \mathbf{e}_{mn} \]

\[ \tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}) = \frac{\exp(\omega(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}))}{\sum_{n'=1}^{N} \exp(\omega(\mathbf{f}_{n'}, \mathbf{q}, \mathbf{g}^{\ell-1}))} \]

\( \omega(\cdot) \) is modeled as a DNN (denoted as DNN\(_1^{(\ell)}\))
Executor->Row Annotator

- More complex information mix than Reader

\[ a_m^\ell = f_A^\ell (r_m^\ell, q, M^{\ell-1}) = DNN_2^{(\ell)} ([r_m^\ell; q; a_m^{\ell-1}; g^{\ell-1}]) \]

I think Reader and Annotator are compensatory to some extent, e.g., a 2-D attention mechanism (see Latent Predictor Network)
Executor->Table Annotator

- Pooling vector

\[ g^\ell = f_{\text{POOL}}(a_1^\ell, a_2^\ell, \ldots, a_M^\ell) = [g_1, g_2, \ldots, g_{d_G}]^\top \]
How is the table annotation used?

5 executors (predefined)

- For intermediate layers (1--4), g is stored in memory, and used when computing the next layer's Reader

\[
\mathbf{r}_m^\ell = f_R(\mathbf{R}_m, \mathbf{F}_\mathcal{T}, \mathbf{q}, \mathcal{M}_m^{\ell-1}) = \sum_{n=1}^{N} \tilde{\omega}(f_n, \mathbf{q}, g_{m-1}^\ell) e_{mn}
\]

\[
\tilde{\omega}(f_n, \mathbf{q}, g_{m-1}^\ell) = \frac{\exp(\omega(f_n, \mathbf{q}, g_{m-1}^\ell))}{\sum_{n' = 1}^{N} \exp(\omega(f_{n'}, \mathbf{q}, g_{m-1}^\ell))}
\]

- For the last layer, g is used to compute a probabilistic distribution over the entire table.

\[
p(w_{mn}|Q, \mathcal{T}) = \frac{\exp(\mathcal{f}_{\text{ANS}}(\mathbf{e}_{mn}, \mathbf{q}, a_{m-1}^\ell, g_{m-1}^\ell))}{\sum_{m' = 1}^{M} \sum_{n' = 1}^{N} \exp(\mathcal{f}_{\text{ANS}}(\mathbf{e}_{m'n'}, \mathbf{q}, a_{m'-1}^\ell, g_{m'-1}^\ell))}
\]
Training Objective

- End-to-end learning (N2N)
- Step-by-step learning (SbS)

\[
\mathcal{L}_{\text{N2N}}(\mathcal{D}) = \sum_{i=1}^{N_D} \log p(y^{(i)} = w_{mn}|Q^{(i)}, T^{(i)})
\]

\[
\mathcal{L}_{\text{SbS}}(\mathcal{D}) = \sum_{i=1}^{N_D} [\log p(y^{(i)} = w_{mn}|Q^{(i)}, T^{(i)}) + \alpha \sum_{\ell=1}^{L} \log \tilde{w}(f_{k,\ell}, \cdot, \cdot)]
\]

Athens (probability distribution over entries)

Executor-5 \ Select host_city of r2

Executor-4 \ Memory Layer-4 \ Find r2 in R with max(#_duration)

Executor-3 \ Memory Layer-3 \ Find row sets R where year < a

Executor-2 \ Memory Layer-2 \ Select year of r1 as a

Executor-1 \ Memory Layer-1 \ Find row r1 where host_city=Beijing
Experimental Setups

- Synthetic dataset containing 4 types of queries generated by templates

N.B. Natural Language with Templates $\iff$ Formal language w/ or w/o ambiguity

Neural Enquirer is a kind of Pseudo Compiling

Our synthetic dataset consists of query-table-answer triples $\{(Q^{(i)}, T^{(i)}, y^{(i)})\}$. To generate such a triple, we first randomly sample a table $T^{(i)}$ of size $10 \times 10$ from a synthetic schema of Olympic Games, which has 10 fields, whose values are drawn from a vocabulary of size 240, with 120 country and city names, and 120 numbers. Figure 5 gives an example table with one row. Next, we generate a query $Q^{(i)}$ using predefined templates associated with its gold-standard answer $y^{(i)}$ on $T^{(i)}$.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Example Queries with Annotated SQL-like Logical Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SELECT_WHERE</strong></td>
<td></td>
</tr>
<tr>
<td>$Q_1$: How many people participated in the game in Beijing?</td>
<td></td>
</tr>
<tr>
<td>$F_1$: <code>select #.participants, where host.city = Beijing</code></td>
<td></td>
</tr>
<tr>
<td>$Q_2$: In which country was the game hosted in 2012?</td>
<td></td>
</tr>
<tr>
<td>$F_2$: <code>select host.country, where year = 2012</code></td>
<td></td>
</tr>
<tr>
<td><strong>SUPERLATIVE</strong></td>
<td></td>
</tr>
<tr>
<td>$Q_3$: When was the lastest game hosted?</td>
<td></td>
</tr>
<tr>
<td>$F_3$: <code>argmax(host.city, year)</code></td>
<td></td>
</tr>
<tr>
<td>$Q_4$: How big is the country which hosted the shortest game?</td>
<td></td>
</tr>
<tr>
<td>$F_4$: <code>argmin(country.size, #.duration)</code></td>
<td></td>
</tr>
<tr>
<td><strong>WHERE_SUPERLATIVE</strong></td>
<td></td>
</tr>
<tr>
<td>$Q_5$: How long is the game with the most medals that has fewer than 3,000 participants?</td>
<td></td>
</tr>
<tr>
<td>$F_5$: <code>where #.participants &lt; 3,000, argmax(#.duration, #.medals)</code></td>
<td></td>
</tr>
<tr>
<td>$Q_6$: How many medals are in the first game after 2008?</td>
<td></td>
</tr>
<tr>
<td>$F_6$: <code>where #.year &gt; 2008, argmin(#.medals, #.year)</code></td>
<td></td>
</tr>
<tr>
<td><strong>NEST</strong></td>
<td></td>
</tr>
<tr>
<td>$Q_7$: Which country hosted the longest game before the game in Athens?</td>
<td></td>
</tr>
<tr>
<td>$F_7$: <code>where year&lt;(select year, where host_city=Athens),argmax(host_country,#.duration)</code></td>
<td></td>
</tr>
<tr>
<td>$Q_8$: How many people watched the earliest game that lasts for more days than the game in 1956?</td>
<td></td>
</tr>
<tr>
<td>$F_8$: <code>where #.duration&lt;(select #.duration,where year=1956),argmin(#.audience,#.year)</code></td>
<td></td>
</tr>
</tbody>
</table>
## Quantitative Results

<table>
<thead>
<tr>
<th></th>
<th>(Baseline)</th>
<th>Mixtured-25K</th>
<th>Mixtured-100K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sempre</td>
<td>N2N</td>
<td>SbS</td>
</tr>
<tr>
<td>Select Where</td>
<td>93.8%</td>
<td>96.2%</td>
<td>99.7%</td>
</tr>
<tr>
<td>Superlative</td>
<td>97.8%</td>
<td>98.9%</td>
<td>99.5%</td>
</tr>
<tr>
<td>Where Superlative</td>
<td>34.8%</td>
<td>80.4%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Nest</td>
<td>34.4%</td>
<td>60.5%</td>
<td>92.1%</td>
</tr>
<tr>
<td>Overall Acc.</td>
<td>65.2%</td>
<td>84.0%</td>
<td>96.4%</td>
</tr>
</tbody>
</table>
Qualitative Analysis

$Q_5$: How long is the game with the most medals that has fewer than 3,000 participants?

Figure 6: Weights visualization of query $Q_5$

- year
- host_city
- #_participants
- #_duration
- #_medals
- #_audience
- host_country
- GDP
- country_size
- population
Q7: Which country hosted the longest game before the game in Athens?

Figure 7: Weights visualization of query $Q_7$

- year
- host_city
- #_participants
- #_duration
- #_medals
- #_audience
- host_country
- GDP
- country_size
- population
Q8: How many people watched the earliest game that lasts for more days than the game in 1956?

Figure 8: Weights visualization of query Q8 (an incorrectly answered query)

- year
- host_city
- _participants
- _duration
- _medals
- _audience
- host_country
- GDP
- country_size
- population
Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning
I: Hello Jack, my name is Chandralekha.

R: Nice to meet you, Chandralekha.

I: This new guy doesn't perform exactly as we expected.

R: What do you mean by "doesn't perform exactly as we expected"?
\[
p(y_t | s_t, y_{t-1}, c_t, M) = p(y_t, g | s_t, y_{t-1}, c_t, M)
\quad + p(y_t, c | s_t, y_{t-1}, c_t, M) \quad (4)
\]

where \(g\) stands for the generate-mode, and \(c\) the copy mode. The probability of the two modes are given respectively by

\[
p(y_t, g | \cdot) = \begin{cases} 
\frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\
0, & y_t \in \mathcal{X} \cap \overline{\mathcal{V}} 
\end{cases} \quad (5)
\]

\[
p(y_t, c | \cdot) = \begin{cases} 
\frac{1}{Z} \sum_{j : x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\
0, & \text{otherwise}
\end{cases} \quad (6)
\]

where \(\psi_g(\cdot)\) and \(\psi_c(\cdot)\) are score functions for generate-mode and copy-mode, respectively, and \(Z\) is the normalization term shared by the two modes, \(Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in \mathcal{X}} e^{\psi_c(x)}\).

\[
p = .5 \left( \frac{\exp(z1)}{\sum \exp(z1')} + \frac{\exp(z2)}{\sum \exp(z2')} \right)
\]

It's very hard to determine which one is better than another.

s: state
M: \{h1, .. ht\}, i.e., source's states
\(c\): input context
\(\text{(w/ attent.)}\)

Cf: conditional probability
Micro avg. vs. Macro avg.

Cf: softmax w/ multiple input
Mean field approximation
\[ p(y_t|s_t, y_{t-1}, c_t, M) = p(y_t, g|s_t, y_{t-1}, c_t, M) \]
\[ + p(y_t, c|s_t, y_{t-1}, c_t, M) \]  

where \( g \) stands for the generator copy mode. The probability of \( y_t \) given respectively by

\[
p(y_t, g|\cdot) = \begin{cases} 
\frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\
0, & y_t \in \mathcal{X} \cap \bar{\mathcal{V}} \\
\frac{1}{Z} e^{\psi_g(\text{UNK})}, & y_t \notin \mathcal{V} \cup \mathcal{X}
\end{cases}
\]  

\[
p(y_t, c|\cdot) = \begin{cases} 
\frac{1}{Z} \sum_{j:x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\
0, & \text{otherwise}
\end{cases}
\]  

where \( \psi_g(\cdot) \) and \( \psi_c(\cdot) \) are score functions for generate-mode and copy-mode, respectively, and \( Z \) is the normalization term shared by the two modes, \( Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in \mathcal{X}} e^{\psi_c(x)} \).
\[ p(y_t|s_t, y_{t-1}, c_t, M) = p(y_t, g_t, c_t) + p(y_t, c|s_t, M) \]

where \( g \) stands for the generate copy mode. The probability of the two given respectively by

\[
p(y_t, g \cdot) = \begin{cases} 
\frac{1}{Z} e^{\psi_g(y_t)}, \\
0, \\
\frac{1}{Z} e^{\psi_g(\text{UNK})}
\end{cases}
\]

\[
p(y_t, c \cdot) = \begin{cases} 
\frac{1}{Z} \sum_{j: x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\
0 & \text{otherwise}
\end{cases}
\]

**Copy-Mode:** The score for “copying” the word \( x_j \) is calculated as

\[
\psi_c(y_t = x_j) = \sigma \left( h_j^T W_c \right) s_t, \quad x_j \in \mathcal{X}
\]

where \( W_c \in \mathbb{R}^{d_h \times d_s} \), and \( \sigma \) is an activation that is either an identity or a non-linear function such as \( \tanh \). When calculating the copy-mode score, we use the hidden states \( \{ h_1, \ldots, h_{T_S} \} \) to “represent” each of the word in the source sequence \( \{ x_1, \ldots, x_{T_S} \} \) since the bi-directional RNN encodes not only the content, but also the location information into the hidden states in \( M \).

where \( \psi_g(\cdot) \) and \( \psi_c(\cdot) \) are score functions for generate-mode and copy-mode, respectively, and \( Z \) is the normalization term shared by the two modes,

\[
Z = \sum_{v \in \mathcal{V} \cup \{ \text{UNK} \}} e^{\psi_g(v)} + \sum_{x \in \mathcal{X}} e^{\psi_c(x)}.
\]
State Update (Input)

- \[ \mathbf{e}(y_{t-1}); \zeta(y_{t-1})^\top \]
- \( \mathbf{e}() \): embedding of a word
- \( \zeta() \):
  \[
  \zeta(y_{t-1}) = \sum_{\tau=1}^{T_S} \rho_{t\tau} \mathbf{h}_\tau
  \]
  If the last word is copied from \( x_t \)

  \[
  \rho_{t\tau} = \begin{cases} 
  \frac{1}{K} p(x_\tau, c|s_{t-1}, \mathbf{M}), & x_\tau = y_{t-1} \\
  0 & \text{otherwise}
  \end{cases}
  \]

  \[
  K = \sum_{\tau': x_{\tau'} = y_{t-1}} p(x_{\tau'}, c|s_{t-1}, \mathbf{M})
  \]

I don't see formal definition of \( \rho \), but it shall be similar to atten.

- \( \mathbf{c}_t = \sum_{\tau=1}^{T_S} \alpha_{t\tau} \mathbf{h}_\tau; \quad \alpha_{t\tau} = \frac{e^{\eta(s_{t-1}, h_\tau)}}{\sum_{\tau'} e^{\eta(s_{t-1}, h_{\tau'})}} \)
State Update (Input)

- \([e(y_{t-1}); \zeta(y_{t-1})]^\top\]
- \(e()\): embedding of a word
- \(\zeta()\):

\[
\zeta(y_{t-1}) = \sum_{\tau = 1}^{T_s} \rho_{t\tau} h_\tau
\]

\[
\rho_{t\tau} = \begin{cases} 
\frac{1}{K} p(x_\tau, c|s_{t-1}, M), & x_\tau = y_{t-1} \\
0 & \text{otherwise}
\end{cases}
\]

\[
K = \sum_{\tau': x_{\tau'} = y_{t-1}} p(x_{\tau'}, c|s_{t-1}, M)
\]

Local-based Addressing (good for OOV)

\[
\zeta(y_{t-1}) \xrightarrow{\text{update}} s_t \xrightarrow{\text{predict}} y_t \xrightarrow{\text{sel. read}} \zeta(y_t)
\]

If the last word is copied from \(x_t\)
Learning

• End-to-end fashion

\[
\mathcal{L} = -\frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \log \left[ p(y_t^{(k)} | y_{<t}^{(k)}, X^{(k)}) \right]
\]
Discussion

- Designing highly (more and more) complicated neural networks to mimic human behaviors: modeling a sentence, querying a table/KB, selecting a field/column, selecting a row, copying something, etc.

- The network has been somewhat over-complicated; it is very hard to judge which part actually contributes to the performance.

- Evaluation is oftentimes weak: synthetic data, subjective evaluation, or criterion not clear (e.g., genQA), etc.

- Nevertheless, an important school of DL4NLP.
A Wider Scope

- Learning to Execute
- Neural Programmer
- Neural Program Interpreter
- Latent Predictor Network for Code Generation

Challenge of end-to-end learning:
- Information processing

<table>
<thead>
<tr>
<th></th>
<th>avg</th>
<th>sum</th>
<th>max</th>
<th>attention</th>
<th>argmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differentiability</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😷</td>
</tr>
<tr>
<td>Supervision</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😷</td>
</tr>
<tr>
<td>Scalability</td>
<td>😷</td>
<td>😷</td>
<td>😷</td>
<td>😷</td>
<td>😊</td>
</tr>
</tbody>
</table>
Intuition

- Using external information to guide an NN instead of designing end-to-end machines
  - Better performance in short term
  - May or may not conform to the goal of AI, depending on how strict the external information is

<table>
<thead>
<tr>
<th></th>
<th>Hard mechanism</th>
<th>(e.g., if-statement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differentiability</td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Supervision</td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>😊</td>
<td></td>
</tr>
</tbody>
</table>