Learning to Compose Neural Networks for Question Answering

Authors: Jacob Andreas, Marcus Rohrbach, Trevor Darrell, Dan Klein

Research Highlight Presented by Zhedi Liu
High Level Overview

A compositional, attentional model for answering questions about a variety of world representations, including images and structured knowledge bases.
Two components, Trained Jointly

- Query: What cities are in Georgia?
- A collection of neural “modules” that can be freely composed
Two components, Trained Jointly

- Query: What cities are in Georgia?
- A network layout predictor that assembles modules into complete deep networks tailored to each question.
Model: Built around Two Distributions

- **A Layout Model:** \( p(z|x; \theta_e) \)
  - chooses a layout for a sentence

- **An Execution Model:** \( p_z(y|w; \theta_e) \)
  - applies the network specified a particular layout to a world representation

1. \( w \) a world representation
2. \( x \) a question
3. \( y \) an answer
4. \( z \) a network layout
5. \( \theta \) a collection of model parameters
Layout Model

Step 1: Represent the input sentence as a dependency tree.

Query: What cities are in Georgia?
Step 2: Associate fragments of the dependency parse with appropriate modules

Query: What cities are in Georgia?
Layout Model

Step 3: Assemble fragments into full layouts

Query: What cities are in Georgia?
Layout Scoring Model

- Produce an LSTM representation of the question, a feature-based representation of the query, and pass both representations through a multilayer perceptron

- The update to the layout-scoring model at each timestep is simply the gradient of the log-probability of the chosen layout, scaled by the accuracy of that layout’s predictions
Query: What cities are in Georgia?
Module: lookup

**Lookup**

(→ **Attention**)

${\text{lookup}[i]}$ produces an attention focused entirely at the index $f(i)$, where the relationship $f$ between words and positions in the input map is known ahead of time (e.g. string matches on database fields).

$$
[\text{lookup}[i]] = e_{f(i)}
$$

(2)

where $e_i$ is the basis vector that is 1 in the $i$th position and 0 elsewhere.
Module: relate

**Relate** \[(\text{Attention} \rightarrow \text{Attention})\]

relate directs focus from one region of the input to another. It behaves much like the find module, but also conditions its behavior on the current region of attention \(h\). Let \(\tilde{w}(h) = \sum_k h_k w^k\), where \(h_k\) is the \(k^{th}\) element of \(h\). Then,

\[
[\text{relate}[i](h)] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus D\tilde{w}(h) \oplus e))
\]  

(4)
Module: find

Find

\[ [\text{find}[i]] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus d)) \] (3)

\( \rightarrow \) Attention

find\([i]\) computes a distribution over indices by concatenating the parameter argument with each position of the input feature map, and passing the concatenated vector through a MLP.
Module: and

And \quad (\text{Attention}^* \rightarrow \text{Attention})

and performs an operation analogous to set intersection for attentions. The analogy to probabilistic logic suggests multiplying probabilities:

$$[\text{and}(h^1, h^2, \ldots)] = h^1 \odot h^2 \odot \cdots$$  \quad (5)
Train an Execution Model

- Maximize \[ \sum_{(w,y,z)} \log p_z(y|w; \theta_e) \]

1. \( w \) a world representation
2. \( x \) a question
3. \( y \) an answer
4. \( z \) a network layout
5. \( \theta \) a collection of model parameters
State-of-the-art Performance: VQA

What is in the sheep’s ear?
(describe[what]
  (and find[sheep]
    find[ear]))
tag

What color is she wearing?
(describe[color]
  find[wear])
white

What is the man dragging?
(describe[what]
  find[man])
boat (board)
State-of-the-art Performance: VQA

<table>
<thead>
<tr>
<th></th>
<th>test-dev</th>
<th>test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes/No</td>
<td>Number</td>
</tr>
<tr>
<td>Zhou (2015)</td>
<td>76.6</td>
<td>35.0</td>
</tr>
<tr>
<td>Noh (2015)</td>
<td>80.7</td>
<td>37.2</td>
</tr>
<tr>
<td>Yang (2015)</td>
<td>79.3</td>
<td>36.6</td>
</tr>
<tr>
<td>NMN</td>
<td>81.2</td>
<td>38.0</td>
</tr>
<tr>
<td>D-NMN</td>
<td>81.1</td>
<td>38.6</td>
</tr>
</tbody>
</table>
State-of-the-art Performance: GeoQA

<table>
<thead>
<tr>
<th>Question</th>
<th>Accuracy</th>
<th>GeoQA</th>
<th>GeoQA+Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Key Largo an island?</td>
<td>yes:</td>
<td>correct</td>
<td></td>
</tr>
<tr>
<td>(exists (and lookup[key-largo] find[island]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What national parks are in Florida?</td>
<td>everglades:</td>
<td>correct</td>
<td></td>
</tr>
<tr>
<td>(and find[park] (relate[in] lookup[florida]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What are some beaches in Florida?</td>
<td>yes (daytona-beach): wrong parse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(exists (and lookup[beach] (relate[in] lookup[florida])))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What beach city is there in Florida?</td>
<td>[none] (daytona-beach): wrong module behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(and lookup[beach] lookup[city] (relate[in] lookup[florida]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>GeoQA</td>
<td>GeoQA+Q</td>
<td></td>
</tr>
<tr>
<td>LSP-F</td>
<td>48</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>LSP-W</td>
<td>51</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>NMN</td>
<td>51.7</td>
<td>35.7</td>
<td></td>
</tr>
<tr>
<td>D-NMN</td>
<td>54.3</td>
<td>42.9</td>
<td></td>
</tr>
</tbody>
</table>