Tracking the World State with Recurrent Entity Networks

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, Yann LeCun

Task

At each timestep, get information (in the form of a sentence) about the state of the world. Then answer a question.

When we get new information, we should update our representation of the world state. The world state can be decomposed into the state of each entity in the world, so we only need to update one entity.

Architecture

The memory model:

- input: a sequence of vectors $s_1, \cdots, s_T$
- output: a set of entity representations $h_1, \cdots h_k$

The world is a collection of entities. Information about each entity is stored in a single cell. Each cell comes with a key and a memory slot.
\[ g_j \leftarrow \sigma(s_t^T h_j + s_t^T w_j) \]
\[ \tilde{h}_j \leftarrow \phi(U h_j + V w_j + W s_t) \]
\[ h_j \leftarrow h_j + g_j \odot \tilde{h}_j \]
\[ h_j \leftarrow \frac{h_j}{||h_j||} \]

multiple-cells at multiple timesteps:
Input Encoder:
- input: a sequence of sentences.
- output: an encoding of each sentence as a fixed sized vector

\[ S_t = \sum_{i} f_i \odot e_i \]

\( e_i \) are pretrained embeddings

Output Module:
- input: a query vector \( q \) and the outputs of the memory model
- output: arbitrary vector (log probabilities over words)
\[ p_j = \text{Softmax}(q^T h_j) \]

\[ u = \sum_j p_j h_j \]

\[ y = R\phi(q + Hu) \]

Look at only one entity and drop the query:

\[ y = R\phi(Hh_j) = R\phi_j \]

\[ y_i = R_i\phi_j \]

Key vectors

- the model should identify entities by keys, which are trainable

\[ g_j \leftarrow \sigma(s_t^T h_j + s_t^T w_j) \]

Key tying:

- Use parser to identify entities.
- One memory cell for each entity.
- Freeze key vector to be word embedding of an entity.

Related work
Tracking the World State with Recurrent Entity Networks

LSTM:
- Forget gate layer:

\[ f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \]

- Input gate layer & tanh(hyperbolic tangent) layer:

\[ i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \]

\[ c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \]

<table>
<thead>
<tr>
<th>Memory Network</th>
<th>RENN</th>
</tr>
</thead>
<tbody>
<tr>
<td>store the entire input sequence in (dynamic long-term) memory</td>
<td>a fixed number of blocks (a window of words) of hidden states as memories</td>
</tr>
<tr>
<td>sequentially update a controller's hidden state via a softmax gating over the memories</td>
<td>update each block with an independent gated RNN</td>
</tr>
</tbody>
</table>

Compared to RENN, CommNN, Interaction Network, Neural Physics Engine use parallel recurrent models without gating mechanism.

Experiments

Synthetic world model task

Task details:
- Two agents randomly placed in a 10x10 grid
- Answer the locations of the agents based on up to T-2 supporting facts

Details:
- 5 memory slots
- 20D per cell
bAbI

Details:

- 20 memory cells
- 100D embedding
- \( U = V = 0, W = I, \phi = \text{identity matrix} \)
Story

mary got the milk there
john moved to the bedroom
sandra went back to the kitchen
mary travelled to the hallway
john got the football there
john went to the hallway
john put down the football
mary went to the garden
john went to the kitchen
sandra travelled to the hallway
daniel went to the hallway
mary discarded the milk
where is the milk?
answer: garden
Interpreting representations

Recall that

\[ y_i = R_i \phi_j \]

Find closest \( R_i \) for each entity \( \phi_j \):
CBT

Input:
1. 20 sentences
2. 21st sentence with missing word
3. list of candidate words

Details:
- Tied keys to candidate words
- Dropout
- $U = V = 0, W = I, \phi = \text{id}$
- No normalization

<table>
<thead>
<tr>
<th>Key</th>
<th>1-NN</th>
<th>2-NN</th>
<th>Story</th>
</tr>
</thead>
<tbody>
<tr>
<td>football</td>
<td>hallway (0.135)</td>
<td>dropped (0.056)</td>
<td>mary got the milk there</td>
</tr>
<tr>
<td>milk</td>
<td>garden (0.111)</td>
<td>took (0.011)</td>
<td>john moved to the bedroom</td>
</tr>
<tr>
<td>john</td>
<td>kitchen (0.501)</td>
<td>dropped (0.027)</td>
<td>sandra went back to the kitchen</td>
</tr>
<tr>
<td>mary</td>
<td>garden (0.442)</td>
<td>took (0.034)</td>
<td>mary travelled to the hallway</td>
</tr>
<tr>
<td>sandra</td>
<td>hallway (0.394)</td>
<td>kitchen (0.121)</td>
<td>john got the football there</td>
</tr>
<tr>
<td>daniel</td>
<td>hallway (0.689)</td>
<td>to (0.076)</td>
<td>john went to the hallway</td>
</tr>
<tr>
<td>bedroom</td>
<td>hallway (0.367)</td>
<td>dropped (0.075)</td>
<td>john put down the football</td>
</tr>
<tr>
<td>kitchen</td>
<td>kitchen (0.483)</td>
<td>daniel (0.029)</td>
<td>mary went to the garden</td>
</tr>
<tr>
<td>garden</td>
<td>garden (0.281)</td>
<td>where (0.026)</td>
<td>john went to the kitchen</td>
</tr>
<tr>
<td>hallway</td>
<td>hallway (0.475)</td>
<td>left (0.060)</td>
<td>sandra travelled to the hallway</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Named Entities</th>
<th>Common Nouns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kneseer-Ney Language Model + cache</td>
<td>0.439</td>
<td>0.577</td>
</tr>
<tr>
<td>LSTMs (context + query)</td>
<td>0.418</td>
<td>0.560</td>
</tr>
<tr>
<td>Window LSTM</td>
<td>0.436</td>
<td>0.582</td>
</tr>
<tr>
<td>EntNet (general)</td>
<td>0.484</td>
<td>0.540</td>
</tr>
<tr>
<td>EntNet (simple)</td>
<td><strong>0.616</strong></td>
<td><strong>0.588</strong></td>
</tr>
<tr>
<td>MemNN</td>
<td>0.493</td>
<td>0.554</td>
</tr>
<tr>
<td>MemNN + self-sup.</td>
<td>0.666</td>
<td>0.630</td>
</tr>
<tr>
<td>Attention Sum Reader (Kadlec et al., 2016)</td>
<td>0.686</td>
<td>0.634</td>
</tr>
<tr>
<td>Gated-Attention Reader (Bhuwan Dhirra &amp; Salakhutdinov, 2016)</td>
<td>0.690</td>
<td>0.639</td>
</tr>
<tr>
<td>EpiReader (Trischler et al., 2016)</td>
<td>0.697</td>
<td>0.674</td>
</tr>
<tr>
<td>AoA Reader (Cui et al., 2016)</td>
<td>0.720</td>
<td>0.694</td>
</tr>
<tr>
<td>NSE Adaptive Computation (Munkhdalai &amp; Yu, 2016)</td>
<td><strong>0.732</strong></td>
<td><strong>0.714</strong></td>
</tr>
</tbody>
</table>