Outline

• Last minute tips for projects

• Model overview and combinations

• Dynamic memory networks
Last minute tips

• Nothing works and everything is too slow → Panic

• Simplify model → Go back to basics: bag of vectors + nnet
• Make a smaller network and dataset for debugging
• Once no bugs: increase model size
• Make sure you can overfit to your dataset
• Plot your training and dev errors over training iterations
• Then regularize with L2 and Dropout
• Then do hyperparameter search

• Come to OH! (}
Model comparison

- **Bag of Vectors**: Surprisingly good baseline for simple text classification problems. Especially if followed by a few relu layers!
- **Window Model**: Good for single word classification for problems that do not need wide context, e.g. POS
- **CNNs**: good for classification, unclear how to incorporate phrase level annotation (can only take a single label), need zero padding for shorter phrases, hard to interpret, easy to parallelize on GPUs, can be very efficient and versatile
- **Recurrent Neural Networks**: Cognitively plausible (reading from left to right, keeping a state), not best for classification (n-gram), slower than CNNs, can do sequence tagging and classification, very active research, amazing with attention mechanisms
- **TreeRNNs**: Linguistically plausible, hard to parallelize, tree structures are discrete and harder to optimize, need a parser
- **Combinations and extensions!**
But, there’s more

• Combine and extend creatively

• Rarely do we use the vanilla models as is
TreeLSTMs

- LSTMs are great
- TreeRNNs can benefit from gates too → TreeRNNs + LSTMs
- Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks by Kai Sheng Tai, Richard Socher, Christopher D. Manning
TreeLSTMs

- Standard LSTM
- Only has one child

\[
\begin{align*}
    i_t &= \sigma \left( W^{(i)} x_t + U^{(i)} h_{t-1} + b^{(i)} \right), \\
    f_t &= \sigma \left( W^{(f)} x_t + U^{(f)} h_{t-1} + b^{(f)} \right), \\
    o_t &= \sigma \left( W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)} \right), \\
    u_t &= \tanh \left( W^{(u)} x_t + U^{(u)} h_{t-1} + b^{(u)} \right), \\
    c_t &= i_t \odot u_t + f_t \odot c_{t-1}, \\
    h_t &= o_t \odot \tanh(c_t),
\end{align*}
\]

TreeLSTM

Has multiple child nodes:

\[
\begin{align*}
    \tilde{h}_j &= \sum_{k \in C(j)} h_k, \\
    i_j &= \sigma \left( W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right), \\
    f_{jk} &= \sigma \left( W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right), \\
    o_j &= \sigma \left( W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right), \\
    u_j &= \tanh \left( W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right), \\
    c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \\
    h_j &= o_j \odot \tanh(c_j),
\end{align*}
\]
RNNs are Slow → Combine with CNNs

• RNNs are the most common basic building block for deepNLP

• Idea: Take the best and parallelizable parts of RNNs and CNNs

• Quasi-Recruent Neural Networks by
  James Bradbury, Stephen Merity, Caiming Xiong & Richard Socher
Quasi-Recurrent Neural Network

- Parallelism computation across time:
  \[ z_t = \tanh(W^1_z x_{t-1} + W^2_z x_t) \quad Z = \tanh(W_z \ast X) \]
  \[ f_t = \sigma(W^1_f x_{t-1} + W^2_f x_t) \quad F = \sigma(W_f \ast X) \]
  \[ o_t = \sigma(W^1_o x_{t-1} + W^2_o x_t) \quad O = \sigma(W_o \ast X) \]

- Element-wise gated recurrence for parallelism across channels:
  \[ h_t = f_t \odot h_{t-1} + (1 - f_t) \odot z_t, \]
Q-RNNs for Language Modeling

- Better

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (medium) (Zaremba et al., 2014)</td>
<td>20M</td>
<td>86.2</td>
<td>82.7</td>
</tr>
<tr>
<td>Variational LSTM (medium) (Gal &amp; Ghahramani, 2016)</td>
<td>20M</td>
<td>81.9</td>
<td>79.7</td>
</tr>
<tr>
<td>LSTM with CharCNN embeddings (Kim et al., 2016)</td>
<td>19M</td>
<td>—</td>
<td>78.9</td>
</tr>
<tr>
<td>Zoneout + Variational LSTM (medium) (Merity et al., 2016)</td>
<td>20M</td>
<td>84.4</td>
<td>80.6</td>
</tr>
</tbody>
</table>

- Our models

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (medium)</td>
<td>20M</td>
<td>85.7</td>
<td>82.0</td>
</tr>
<tr>
<td>QRNN (medium)</td>
<td>18M</td>
<td>82.9</td>
<td>79.9</td>
</tr>
<tr>
<td>QRNN + zoneout ($p = 0.1$) (medium)</td>
<td>18M</td>
<td>82.1</td>
<td>78.3</td>
</tr>
</tbody>
</table>

- Faster

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Sequence length</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5.5x</td>
<td>8.8x</td>
<td>11.0x</td>
<td>12.4x</td>
<td>16.9x</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5.5x</td>
<td>6.7x</td>
<td>7.8x</td>
<td>8.3x</td>
<td>10.8x</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4.2x</td>
<td>4.5x</td>
<td>4.9x</td>
<td>4.9x</td>
<td>6.4x</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3.0x</td>
<td>3.0x</td>
<td>3.0x</td>
<td>3.0x</td>
<td>3.7x</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>2.1x</td>
<td>1.9x</td>
<td>2.0x</td>
<td>2.0x</td>
<td>2.4x</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>1.4x</td>
<td>1.4x</td>
<td>1.3x</td>
<td>1.3x</td>
<td>1.3x</td>
<td></td>
</tr>
</tbody>
</table>
Q-RNNs for Sentiment Analysis

- Often better and faster than LSTMs
- More interpretable
- Example:
  - Initial positive review
  - Review starts out positive
  - At 117: “not exactly a bad story”
  - At 158: “I recommend this movie to everyone, even if you’ve never played the game”
Neural Architecture Search!

• Manual process of finding best units requires a lot of expertise

• What if we could use AI to find the right architecture for any problem?

• Neural architecture search with reinforcement learning by Zoph and Le, 2016
Neural Architecture Search

Sample architecture $A$ with probability $p$

The controller (RNN)

Trains a child network with architecture $A$ to get accuracy $R$

Compute gradient of $p$ and scale it by $R$ to update the controller
Example: CNN Controller

Used Reinforcement Learning to train the RNN Controller
LSTM Cell vs NAS Cell
## Nice Perplexity Reduction for Language Modeling

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Test Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mikolov &amp; Zweig (2012) - KN-5</td>
<td>2M</td>
<td>141.2</td>
</tr>
<tr>
<td>Mikolov &amp; Zweig (2012) - KN5 + cache</td>
<td>2M</td>
<td>125.7</td>
</tr>
<tr>
<td>Mikolov &amp; Zweig (2012) - RNN</td>
<td>6M</td>
<td>124.7</td>
</tr>
<tr>
<td>Mikolov &amp; Zweig (2012) - RNN-LDA</td>
<td>7M</td>
<td>113.7</td>
</tr>
<tr>
<td>Mikolov &amp; Zweig (2012) - RNN-LDA + KN-5 + cache</td>
<td>9M</td>
<td>92.0</td>
</tr>
<tr>
<td>Pascaru et al. (2013) - Deep RNN</td>
<td>6M</td>
<td>107.5</td>
</tr>
<tr>
<td>Cheng et al. (2014) - Sum-Prod Net</td>
<td>5M</td>
<td>100.0</td>
</tr>
<tr>
<td>Zaremba et al. (2014) - LSTM (medium)</td>
<td>20M</td>
<td>82.7</td>
</tr>
<tr>
<td>Zaremba et al. (2014) - LSTM (large)</td>
<td>66M</td>
<td>78.4</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (medium, untied)</td>
<td>20M</td>
<td>79.7</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (medium, untied, MC)</td>
<td>20M</td>
<td>78.6</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (large, untied)</td>
<td>66M</td>
<td>75.2</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (large, untied, MC)</td>
<td>66M</td>
<td>73.4</td>
</tr>
<tr>
<td>Kim et al. (2015) - CharCNN</td>
<td>19M</td>
<td>78.9</td>
</tr>
<tr>
<td>Press &amp; Wolf (2016) - Variational LSTM, shared embeddings</td>
<td>51M</td>
<td>73.2</td>
</tr>
<tr>
<td>Merity et al. (2016) - Zoneout + Variational LSTM (medium)</td>
<td>20M</td>
<td>80.6</td>
</tr>
<tr>
<td>Merity et al. (2016) - Pointer Sentinel-LSTM (medium)</td>
<td>21M</td>
<td>70.9</td>
</tr>
<tr>
<td>Inan et al. (2016) - VD-LSTM + REAL (large)</td>
<td>51M</td>
<td>68.5</td>
</tr>
<tr>
<td>Zilly et al. (2016) - Variational RHN, shared embeddings</td>
<td>24M</td>
<td>66.0</td>
</tr>
<tr>
<td>Neural Architecture Search with base 8</td>
<td>32M</td>
<td>67.9</td>
</tr>
<tr>
<td>Neural Architecture Search with base 8 and shared embeddings</td>
<td>25M</td>
<td>64.0</td>
</tr>
<tr>
<td>Neural Architecture Search with base 8 and shared embeddings</td>
<td>54M</td>
<td>62.4</td>
</tr>
</tbody>
</table>
More complex tasks need more complex architectures

• So far, we looked at basic sequence models and seq2seq models

• As you know from the default final project, some tasks require more complex memory components

• One of the first ones that was shown to work on both synthetic problems and real NLP tasks:

• Dynamic Memory Networks by Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, Richard Socher
High level idea for harder questions

- Imagine having to read an article, memorize it, then get asked various questions → Hard!
- You can't store everything in working memory
- **Optimal**: give you the input data, give you the question, allow as many glances as possible
For datasets that mark which facts are important for a given question, such as Facebook’s bAbI, it is described here as its own computation to highlight the potential modularity of these subcomponents.

Finally, to summarize the episode for pass $i$, we employ a modified GRU over the sequence of candidate facts $c_i$, and these episodes are then summarized into the memory. Endowing our episodic component allows its attention mechanism to attend more selectively to specific facts on each pass, as it can attend to other important facts at a later pass. It also allows for a type of transitive inference, since the first pass may uncover the need to retrieve additional facts.

In its general form, the episodic memory module is characterized by an attention mechanism, a gating function, which takes as input, for each pass $i$, the input module, and a function that summarizes the episodes into a memory.

In our work, we use a gating function as our attention mechanism. It takes as input, for each pass $i$, the memory $m$, the question $q$, and the candidate fact $e_i$.

$$
G(c_i, m, q) = \tanh(c_i \cdot W + m \cdot V + q \cdot U + b)
$$

A gating function returns a single scalar and is defined as

$$
G(c_i, m, q) = \frac{1}{1 + \exp(-G(c_i, m, q))}
$$

The state is updated by way of a GRU:

$$
\begin{align*}
    h_i &= GRU(c_i, h_{i-1}) \\
    m_i &= G(h_i, m_{i-1}, q) m_{i-1} + (1 - G(h_i, m_{i-1}, q)) m_{i-1}
\end{align*}
$$

where $m_0$ is the computed episode at pass $0$, and we set the memory $m_0$ to simply the attention mechanism's final state, but we have de-

The question vector itself helps; e.g., John put down the football. Only once the model sees that John is relevant can it reason the second iteration should retrieve where John was. In this example, taken from a true test question on Facebook’s bAbI task, this behavior is indeed seen. Note that the second iteration has wrongly placed some weight in football. The episode is the final state of the GRU:

$$
E = \text{GRU}(c, h_{i-1})
$$

Finally, to take multiple passes over the facts, focusing attention on different facts at each pass. Each pass

$T = 8$

produces an $1 \times 8$ tensor, which is used to compute the answer.

$\text{Answer} = \text{Sum}(E)$

Figure 3: Real example of an input sentence sequence and the attention gates that are triggered by a question vector $q$. The gates change with

$T = 8$
The Modules: Input

Standard GRU. The last hidden state of each sentence is accessible.
Further Improvement: BiGRU

The input module with a "fusion layer", where the sentence reader encodes the sentence and the bi-directional GRU algorithm allows information to flow between sentences.

**Textual Input Module**

- **Facts**
  - GRU
  - GRU
  - GRU

- **Input fusion layer**
  - GRU
  - GRU
  - GRU

- **Sentence reader**
  - Positional Encoder
  - Positional Encoder
  - Positional Encoder

**Visual Input Module**

- **Visual feature embedding**
- **Feature extraction**
- **Visual feature**

**Input Module for VQA**

To apply the DMN to visual question answering, we introduced an image into small local regions and considers each region equivalent to a sentence in the input module for text. To reduce a new input module for images, the module splits an image into small local regions and considers each region equivalent to a sentence in the input module for text. This allows contextual power is quite limited, with simple issues like object scaling or locational variance causing accuracy problems. Without global information, their representational power is quite limited, with simple issues like object scaling or locational variance causing accuracy problems.

**Figure 2.**

**Input fusion layer**

- The input fusion layer takes these input facts and enables the local regional vectors extracted from them. Without global information, their representational power is quite limited, with simple issues like object scaling or locational variance causing accuracy problems.

**Figure 3.**

- The local regional vectors extracted from them. Without global information, their representational power is quite limited, with simple issues like object scaling or locational variance causing accuracy problems.

**Further Improvement: BiGRU**

- GRUs and LSTMs were also considered for simplicity and speed, we selected the positional encoding scheme described in Sukhbaatar et al. (2015) to allow for a comparison to their work. GRUs and LSTMs were also considered for simplicity and speed, we selected the positional encoding scheme described in Sukhbaatar et al. (2015) to allow for a comparison to their work. GRUs and LSTMs were also considered for simplicity and speed, we selected the positional encoding scheme described in Sukhbaatar et al. (2015) to allow for a comparison to their work. GRUs and LSTMs were also considered for simplicity and speed, we selected the positional encoding scheme described in Sukhbaatar et al. (2015) to allow for a comparison to their work.
The Modules: Question

\[ q_t = GRU(v_t, q_{t-1}). \]
Finally, to summarize the episode is the final state of the GRU: the state is updated by way of a GRU:

$$h^t_i = g^t_i GRU(s_i, h^t_{i-1}) + (1 - g^t_i) h^t_{i-1}$$

**Last hidden state:** $m^t$
The Modules: Episodic Memory

- Gates are activated if sentence relevant to the question or memory

\[ z_i^t = [s_i \circ q ; s_i \circ m^{t-1}; |s_i - q| ; |s_i - m^{t-1}|] \]

\[ Z_i^t = W^{(2)} \tanh \left( W^{(1)} z_i^t + b^{(1)} \right) + b^{(2)} \]

- When summ
Finally, to summarize the episode for pass $i$, we employ a modified GRU over the sequence of facts $s_i$. The function $g(x)$ returns a single scalar and is defined as

$$g(x) = \frac{1}{1 + e^{-x}}$$

The state may be initialized randomly, but in practice we have found that initializing it to the average of the input embeddings works well.

If summary is insufficient to answer the question, repeat sequence over input.
The Modules: Answer

\[ a_t = \text{GRU}(y_{t-1}, q, a_{t-1}), \quad y_t = \text{softmax}(W^{(a)} a_t) \]
Related work

- Sequence to Sequence (Sutskever et al. 2014)
- Neural Turing Machines (Graves et al. 2014)
- Teaching Machines to Read and Comprehend (Hermann et al. 2015)
- Learning to Transduce with Unbounded Memory (Grefenstette 2015)
- Structured Memory for Neural Turing Machines (Wei Zhang 2015)
- Memory Networks (Weston et al. 2015)
- End to end memory networks (Sukhbaatar et al. 2015)
Comparison to MemNets

Similarities:
• MemNets and DMNs have input, scoring, attention and response mechanisms

Differences:
• For input representations MemNets use bag of word, nonlinear or linear embeddings that explicitly encode position
• MemNets iteratively run functions for attention and response

• DMNs show that neural sequence models can be used for input representation, attention and response mechanisms → naturally captures position and temporality
• Enables broader range of applications
The Facebook bAbI dataset is a synthetic dataset meant to test a model's ability to retrieve facts and reason over them. Each task tests a different skill that a good question answering model ought to have, such as coreference resolution, deduction, and induction.

<table>
<thead>
<tr>
<th>Task</th>
<th>MemNN</th>
<th>DMN</th>
<th>Task</th>
<th>MemNN</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Single Supporting Fact</td>
<td>100</td>
<td>100</td>
<td>11: Basic Coreference</td>
<td>100</td>
<td>99.9</td>
</tr>
<tr>
<td>2: Two Supporting Facts</td>
<td>100</td>
<td>98.2</td>
<td>12: Conjunction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3: Three Supporting facts</td>
<td>100</td>
<td>95.2</td>
<td>13: Compound Coreference</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td>4: Two Argument Relations</td>
<td>100</td>
<td>100</td>
<td>14: Time Reasoning</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>5: Three Argument Relations</td>
<td>98</td>
<td>99.3</td>
<td>15: Basic Deduction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6: Yes/No Questions</td>
<td>100</td>
<td>100</td>
<td>16: Basic Induction</td>
<td>100</td>
<td>99.4</td>
</tr>
<tr>
<td>7: Counting</td>
<td>85</td>
<td>96.9</td>
<td>17: Positional Reasoning</td>
<td>65</td>
<td>59.6</td>
</tr>
<tr>
<td>8: Lists/Sets</td>
<td>91</td>
<td>96.5</td>
<td>18: Size Reasoning</td>
<td>95</td>
<td>95.3</td>
</tr>
<tr>
<td>9: Simple Negation</td>
<td>100</td>
<td>100</td>
<td>19: Path Finding</td>
<td>36</td>
<td>34.5</td>
</tr>
<tr>
<td>10: Indefinite Knowledge</td>
<td>98</td>
<td>97.5</td>
<td>20: Agent’s Motivations</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Mean Accuracy (%)</td>
<td>93.3</td>
<td>93.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experiments: Sentiment Analysis

Stanford Sentiment Treebank

Test accuracies:
- MV-RNN and RNTN: Socher et al. (2013)
- DCNN: Kalchbrenner et al. (2014)
- PVec: Le & Mikolov. (2014)
- CNN-MC: Kim (2014)
- CT-LSTM: Tai et al. (2015)
- CT-LSTM: Tai et al. (2015)

<table>
<thead>
<tr>
<th>Task</th>
<th>Binary</th>
<th>Fine-grained</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV-RNN</td>
<td>82.9</td>
<td>44.4</td>
</tr>
<tr>
<td>RNTN</td>
<td>85.4</td>
<td>45.7</td>
</tr>
<tr>
<td>DCNN</td>
<td>86.8</td>
<td>48.5</td>
</tr>
<tr>
<td>PVec</td>
<td>87.8</td>
<td>48.7</td>
</tr>
<tr>
<td>CNN-MC</td>
<td>88.1</td>
<td>47.4</td>
</tr>
<tr>
<td>DRNN</td>
<td>86.6</td>
<td>49.8</td>
</tr>
<tr>
<td>CT-LSTM</td>
<td>88.0</td>
<td>51.0</td>
</tr>
<tr>
<td>DMN</td>
<td><strong>88.6</strong></td>
<td><strong>52.1</strong></td>
</tr>
</tbody>
</table>
How many attention + memory passes are needed in the episodic memory?

<table>
<thead>
<tr>
<th>Max passes</th>
<th>task 3 three-facts</th>
<th>task 7 count</th>
<th>task 8 lists/sets</th>
<th>sentiment (fine grain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 pass</td>
<td>0</td>
<td>48.8</td>
<td>33.6</td>
<td>50.0</td>
</tr>
<tr>
<td>1 pass</td>
<td>0</td>
<td>48.8</td>
<td>54.0</td>
<td>51.5</td>
</tr>
<tr>
<td>2 pass</td>
<td>16.7</td>
<td>49.1</td>
<td>55.6</td>
<td>52.1</td>
</tr>
<tr>
<td>3 pass</td>
<td>64.7</td>
<td>83.4</td>
<td>83.4</td>
<td>50.1</td>
</tr>
<tr>
<td>5 pass</td>
<td><strong>95.2</strong></td>
<td><strong>96.9</strong></td>
<td><strong>96.5</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>
Analysis of Attention for Sentiment

- Sharper attention when 2 passes are allowed.
- Examples that are wrong with just one pass

1-iter DMN (pred: negative, ans: positive)

2-iter DMN (pred: positive, ans: positive)
Analysis of Attention for Sentiment

1-iter DMN (pred: very positive, ans: negative)

2-iter DMN (pred: negative, ans: negative)
Analysis of Attention for Sentiment

- Examples where full sentence context from first pass changes attention to words more relevant for final prediction

![Diagrams showing attention weights for sentiment examples with one and two episodes.](image-url)
Examples where full sentence context from first pass changes attention to words more relevant for final prediction

1-iter DMN (pred: positive, ans: negative)

2-iter DMN (pred: negative, ans: negative)
4.1 Question Answering

The Facebook bAbI dataset is a synthetic dataset meant to test a model's ability to retrieve facts and reason over them. Each task tests a different skill that a good question answering model ought to have, such as coreference resolution, deduction, and induction. Training on the bAbI dataset...

<table>
<thead>
<tr>
<th>Task</th>
<th>MemNN</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Single Supporting Fact</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10: Indefinite Knowledge</td>
<td>98</td>
<td>97.5</td>
</tr>
<tr>
<td>11: Basic Coreference</td>
<td>99.9</td>
<td>100</td>
</tr>
<tr>
<td>2: Two Supporting Facts</td>
<td>98.2</td>
<td>100</td>
</tr>
<tr>
<td>12: Conjunction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3: Three Supporting Facts</td>
<td>95.2</td>
<td>99.8</td>
</tr>
<tr>
<td>13: Compound Coreference</td>
<td>99.8</td>
<td>100</td>
</tr>
<tr>
<td>4: Two Argument Relations</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>14: Time Reasoning</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>5: Three Argument Relations</td>
<td>99.3</td>
<td>99.5</td>
</tr>
<tr>
<td>15: Basic Deduction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6: Yes/No Questions</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>16: Basic Induction</td>
<td>99.4</td>
<td>99.4</td>
</tr>
<tr>
<td>7: Counting</td>
<td>85</td>
<td>96.9</td>
</tr>
<tr>
<td>17: Positional Reasoning</td>
<td>65</td>
<td>59.6</td>
</tr>
<tr>
<td>8: Lists/Sets</td>
<td>91</td>
<td>96.5</td>
</tr>
<tr>
<td>18: Size Reasoning</td>
<td>95</td>
<td>95.3</td>
</tr>
<tr>
<td>9: Simple Negation</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>19: Path Finding</td>
<td>36</td>
<td>34.5</td>
</tr>
<tr>
<td>10: Indefinite Knowledge</td>
<td>97.5</td>
<td>97.5</td>
</tr>
<tr>
<td>20: Agent's Motivations</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Mean Accuracy (%) 93.3

Table 1: Test accuracies on the bAbI dataset. MemNN numbers taken from Weston et al. [18]. The DMN passes (accuracy > 95%) 18 tasks, whereas the MemNN passes 16.

In practice, we begin training with \( \beta \) set to 1 and \( \alpha \) set to 0, and then later switch \( \alpha \) to 1 while keeping \( \beta \) at 1. We subsample the facts from the input module by end-of-sentence tokens. The gate supervision aims to select one sentence per pass; thus, we also experimented with modifying Eq. 6 to a simple softmax instead of a GRU.

Here, we compute the final episode vector via:

\[
e_i = \sum_{t=1}^{T} \text{softmax}(g_i t) c_t,
\]

where \( \text{softmax}(g_i t) = \frac{\exp(g_i t)}{\sum_{j=1}^{T} \exp(g_i j)} \), and \( g_i t \) here is the value of the gate before the sigmoid. This setting achieves better results, likely because the softmax is better suited to picking one sentence at a time.

We list results in table 1. The DMN does worse than the MemNN on tasks 2 and 3, both tasks with long input sequences. We suspect this is due to the recurrent input sequence model having trouble modeling very long inputs. The MemNN does not suffer from this problem as it views each sentence separately. The power of the episodic memory module is evident in tasks 7 and 8, where the DMN significantly outperforms the MemNN. Both tasks require the model to iteratively retrieve facts and store them in a representation that slowly incorporates more of the relevant information of the input sequence. Both models do poorly on tasks 17 and 19, though the MemNN does better. We suspect this is due to the MemNN using n-gram features as well as explicit sequence position features.

4.2 Sequence Tagging: Part of Speech Tagging

Part-of-speech tagging is traditionally modeled as a sequence tagging problem: every word in a sentence is to be classified into its part-of-speech class (see Fig. 1). We evaluate on the standard Wall Street Journal dataset included in Penn-III [26]. We use the standard splits of sections 0-18 for training, 19-21 for development and 22-24 for test sets [27]. Since this is a word level tagging task, DMN memories are produced at the word -rather than sentence- level. We compare the DMN...
Modularization Allows for Different Inputs

Input Module

John moved to the garden.
John got the apple there.
John moved to the kitchen.
Sandra picked up the milk there.
John dropped the apple.
John moved to the office.

Question

Where is the apple?

Input Module

What kind of tree is in the background?

Answer

Kitchen

Answer

Palm

Episodic Memory

Episodic Memory

Figure 1. Question Answering over text and images using a Dynamic Memory Network (DMN) is one example of a neural network model that has both a memory component and an attention mechanism. The DMN yields state of the art results on question answering with supporting facts labeled during training, sentiment analysis, and part-of-speech tagging. Its main idea is to use a question to selectively pay attention to textual inputs. These inputs are then given to an episodic memory module which collects the relevant inputs in order to give an answer. The memory module has two important steps: (1) computing attention scores to focus on particular facts given a question and (2) updating the memory by reasoning over the attended facts.

We analyze the DMN components, specifically the input module and memory module, to improve accuracy over question answering. We propose a new input module which uses a two level encoder with a sentence reader and input fusion layer to allow for information flow between sentences. For the memory, we propose a modification to gated recurrent units (GRU). The gates in the new GRU formulation are dependent on the attention scores and global knowledge over the facts. Unlike before, the new DMN+ model does not require that supporting facts (i.e. the facts that are relevant for answering a particular question) are labeled during training. The model learns to pick the important facts from a larger set. In addition, we introduce a new input module to represent images. This module is compatible with the rest of the DMN architecture and its output is fed into the memory module. We show that the changes in the memory module
Input Module for Images

Dynamic Memory Networks for Visual and Textual Question Answering, Caiming Xiong, Stephen Merity, Richard Socher
Accuracy: Visual Question Answering

VQA test-dev and test-standard:
- Antol et al. (2015)
- ACK Wu et al. (2015);
- iBOWIMG - Zhou et al. (2015);
- DPPnet - Noh et al. (2015); D-NMN - Andreas et al. (2016);
- SAN - Yang et al. (2015)

<table>
<thead>
<tr>
<th>Method</th>
<th>test-dev All</th>
<th>Y/N</th>
<th>Other</th>
<th>Num</th>
<th>test-std All</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>28.1</td>
<td>64.0</td>
<td>3.8</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>48.1</td>
<td>75.7</td>
<td>27.1</td>
<td>36.7</td>
<td></td>
</tr>
<tr>
<td>Q+I</td>
<td>52.6</td>
<td>75.6</td>
<td>37.4</td>
<td>33.7</td>
<td></td>
</tr>
<tr>
<td>LSTM Q+I</td>
<td>53.7</td>
<td>78.9</td>
<td>36.4</td>
<td>35.2</td>
<td>54.1</td>
</tr>
<tr>
<td>ACK</td>
<td>55.7</td>
<td>79.2</td>
<td>40.1</td>
<td>36.1</td>
<td>56.0</td>
</tr>
<tr>
<td>iBOWIMG</td>
<td>55.7</td>
<td>76.5</td>
<td>42.6</td>
<td>35.0</td>
<td>55.9</td>
</tr>
<tr>
<td>DPPnet</td>
<td>57.2</td>
<td>80.7</td>
<td>41.7</td>
<td>37.2</td>
<td>57.4</td>
</tr>
<tr>
<td>D-NMN</td>
<td>57.9</td>
<td>80.5</td>
<td>43.1</td>
<td>37.4</td>
<td>58.0</td>
</tr>
<tr>
<td>SAN</td>
<td>58.7</td>
<td>79.3</td>
<td>46.1</td>
<td>36.6</td>
<td>58.9</td>
</tr>
<tr>
<td>DMN+</td>
<td><strong>60.3</strong></td>
<td>80.5</td>
<td>48.3</td>
<td>36.8</td>
<td><strong>60.4</strong></td>
</tr>
</tbody>
</table>
Attention Visualization

What is the main color on the bus? Answer: blue
What type of trees are in the background? Answer: pine
How many pink flags are there? Answer: 2
Is this in the wild? Answer: no
Attention Visualization

Which man is dressed more flamboyantly?  Answer: **right**

Who is on both photos?  Answer: **girl**

What time of day was this picture taken?  Answer: **night**

What is the boy holding?  Answer: **surfboard**
Attention Visualization

What is this sculpture made out of? Answer: metal

What color are the bananas? Answer: green

What is the pattern on the cat's fur on its tail? Answer: stripes

Did the player hit the ball? Answer: yes
What is the girl holding?  tennis racket
What is the girl doing?  playing tennis
Is the girl wearing a hat?  yes
What is the girl wearing?  shorts
What is the color of the ground?  brown
What color is the ball?  yellow
What color is her skirt?  white
What did the girl just hit?  tennis ball
Summary

- Basic blocks can be combined or learned with NAS
- Memory is useful. DMN accurately solves variety of tasks
- Next week: Most recent research and fun future outlook