Outline

Introduction

MemNN: Memory Networks
  Memory Networks: General Framework
  MemNNs for Text
  Experiments

MemNN-WSH: Weakly Supervised Memory Networks
  Introduction
  MemNN-WSH: Memory via Multiple Layers
  Experiments
Outline

Introduction

MemNN: Memory Networks

MemNN-WSH: Weakly Supervised Memory Networks
Authors

- **Memory Networks**
  - **Jason Weston**, Sumit Chopra & Antoine Bordes
  - Facebook AI Research

- **Weakly Supervised Memory Networks**
  - Sainbayar Sukhbaatar, Arthur Szlam, **Jason Weston**, Rob Fergus
  - New York University, Facebook AI Research

Introduction

Recall some toy tasks of Question Answering $^1$:

John is in the playground.
Bob is in the office.
Where is John? A: \textit{playground}

John is in the playground.
Bob is in the office.
John picked up the football.
Bob went to the kitchen.
Where is the football? A: \textit{playground}
Where was Bob before the kitchen? A: \textit{office}

John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A: \textit{office}

Simulated World QA:

- 4 characters, 3 objects and 5 rooms
- characters: moving around, picking up and dropping objects

→ A story, a related question and an answer
Simulated World QA:
- 4 characters, 3 objects and 5 rooms
- characters: moving around, picking up and dropping objects
  → A story, a related question and an answer

To answer the question:
- Understanding the question and the story
- Finding the supporting facts for the question
- Generating an answer based on supporting facts
Introduction (cont.)

Classical QA methods:

• Retrieval based methods:
  Finding answers from a set of documents
• Triple-KB based methods:
  Mapping questions to logical queries
  Querying the knowledge base to find answer related triples

Neural network and embedding approaches:

1. Representing questions and answers as embeddings via neural sentence models
2. Learning matching models and embeddings by question-answer pairs
Classical QA methods:

- Retrieval based methods:
  Finding answers from a set of documents
- Triple-KB based methods:
  Mapping questions to logical queries
  Querying the knowledge base to find answer related triples

Neural network and embedding approaches:

1. Representing questions and answers as embeddings via neural sentence models
2. Learning matching models and embeddings by question-answer pairs

How about reasoning?
Introduction (cont.)

Classical QA methods:

- Retrieval based methods:
  Finding answers from a set of documents

- Triple-KB based methods:
  Mapping questions to logical queries
  Querying the knowledge base to find answer related triples

Neural network and embedding approaches:

1. Representing questions and answers as embeddings via neural sentence models
2. Learning matching models and embeddings by question-answer pairs

**How about reasoning?**

Memory Networks: Reason with inference components combined with a long-term memory component
Outline

Introduction

MemNN: Memory Networks
  Memory Networks: General Framework
  MemNNs for Text
  Experiments

MemNN-WSH: Weakly Supervised Memory Networks
Memory Networks: General Framework

Components: \((m, I, G, O, R)\)

- A memory \(m\): an array of objects indexed by \(m_i\)
- Four (potentially learned) components \(I, G, O \) and \(R\):
  
  - \(I\) – input feature map: converts the incoming input to the internal feature representation.
  
  - \(G\) – generalization: updates old memories given the new input.
  
  - \(O\) – output feature map: produces a new output\(^2\), given the new input and the current memory state.
  
  - \(R\) – response: converts the output into the response format desired.

\(^2\)the output in the feature representation space

\(^3\)Input: e.g., an character, word or sentence, or image or an audio signal
Memory Networks: General Framework

Components: \((m, I, G, O, R)\)
- A memory \(m\): an array of objects indexed by \(m_i\)
- Four (potentially learned) components \(I, G, O\) and \(R\):
  - \(I\) – input feature map: converts the incoming input to the internal feature representation.
  - \(G\) – generalization: updates old memories given the new input.
  - \(O\) – output feature map: produces a new output\(^2\), given the new input and the current memory state.
  - \(R\) – response: converts the output into the response format desired.

Given an input \(x\), the flow of the model\(^3\):

1. Convert \(x\) to an internal feature representation \(I(x)\).
2. Update memories \(m_i\) given the new input: \(m_i = G(m_i, I(x), m), \forall i\).
3. Compute output features \(o\) given the new input and the memory: \(o = O(I(x), m)\).
4. Finally, decode output features \(o\) to give the final response: \(r = R(o)\).

\(^2\) the output in the feature representation space
\(^3\) Input: e.g., an character, word or sentence, or image or an audio signal
Memory Networks: General Framework (cont.)

Memory networks cover a wide class of possible implementations.
The components $I$, $G$, $O$ and $R$ can potentially use any existing ideas from the machine learning literature.

- **$I$**: standard pre-processing or encoding the input into an internal feature representation
- **$G$**: updating memories
  - Simplest form: to store $I(x)$ in a slot in the memory $m_H(x) = I(x)$
  - More sophisticated form: go back and update earlier stored memories based on the new evidence from the current input
- **$O$**: reading from memory and performing inference (e.g., calculating what are the relevant memories to perform a good response)
- **$R$**: producing the final response given $O$ (e.g., embeddings → actual words)

---

4 similar to LSTM
Memory Networks: General Framework (cont.)

Memory networks cover a wide class of possible implementations. The components $I$, $G$, $O$ and $R$ can potentially use any existing ideas from the machine learning literature.

- $I$: standard pre-processing or encoding the input into an internal feature representation
- $G$: updating memories
  - Simplest form: to store $I(x)$ in a slot in the memory $m_H(x) = I(x)$
  - More sophisticated form: go back and update earlier stored memories based on the new evidence from the current input
- Memory is huge (e.g. Freebase): slot choosing functions $H$
- Memory is full/overflowed: implementing a "forgetting" procedure via $H$ to replace memory slots
- $O$: reading from memory and performing inference (e.g., calculating what are the relevant memories to perform a good response)
- $R$: producing the final response given $O$ (e.g., embeddings $\rightarrow$ actual words)

One particular instantiation of a memory network:
- Memory neural networks (MemNNs): the components are neural networks

\[^4\text{similar to LSTM}\]
MemNN Models for Text

Basic MemNN Model for Text:

- \((m, I, G, O, R)\)

Variants of Basic MemNN Model for Text

- Word Sequences as Input
- Efficient Memory via Hashing
- Modeling Writing Time
- Modeling Previous Unseen Words
- Exact Matches and Unseen Words
MemNNs for Text: Basic Model

- **I**: input text— a sentence (the statement of a fact, or a question)
- **G**: storing text in the next available memory slot in its original form:
  \[ m_N = x, N = N + 1 \]
  
  \( G \) only used to store new memory, old memories are not updated.
- **O**: producing output features by finding \( k \) supporting memories given \( x \)
  Take \( k = 2 \) as an example:
  \[
  o_1 = O_1(x, m) = \arg \max_{i=1,\ldots,N} s_O(x, m_i) \\
  o_2 = O_2(x, m) = \arg \max_{i=1,\ldots,N} s_O([x, m_{o1}], m_i)
  \]
  
  The final output \( o \): \([x, m_{o1}, m_{o2}]\)
- **R**: producing a textual response \( r \)
  \[
  r = \arg \max_{w \in W} s_R([x, m_{o1}, m_{o2}], w)
  \]
  where \( W \) is the word vocabulary

```
Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk. 
Joe travelled to the office. Joe left the milk. Joe went to the bathroom. 
Where is the milk now? A: office 
Where is Joe? A: bathroom 
Where was Joe before the office? A: kitchen
```
MemNNs for Text: Basic Model (cont.)

Scoring Function for the output and response: $s_O, s_R$

$$s(x, y) = \Phi_x(x)^T U^T U \Phi_y(y)$$

where for $s_O$: $x$ — input and supporting memory, $y$ — next supporting memory

for $s_R$: $x$ — output in the feature space, $y$ — actual response (words or phrases)

$U \in \mathbb{R}^{n \times D}(U_O, U_R)$, $n$ : the embedding dimension, $D$ : the number of features

$\Phi_x, \Phi_y$ : mapping the original text to the $D$-dimensional feature representation

$D = 3|W|$, one for $\Phi_y(\cdot)$, two for $\Phi_x(\cdot)$ (input from $x$ or $m$)
MemNNs for Text: Basic Model (cont.)

Scoring Function for the output and response: $s_O, s_R$

$$s(x, y) = \Phi_x(x)^T U^T U \Phi_y(y)$$

where for $s_O$: $x$ — input and supporting memory, $y$ — next supporting memory

for $s_R$: $x$ — output in the feature space, $y$ — actual response (words or phrases)

$U \in \mathbb{R}^{n \times D}(U_O, U_R)$, $n$: the embedding dimension, $D$: the number of features

$\Phi_x, \Phi_y$: mapping the original text to the $D$-dimensional feature representation

$D = 3|W|$, one for $\Phi_y(\cdot)$, two for $\Phi_x(\cdot)$ (input from $x$ or m)

Training: a fully (or strongly) supervised setting

- labeled: inputs and responses, and the supporting sentences (in all steps)
- objective function: a margin ranking loss

For a given question $x$ with true response $r$ and supporting sentences $f_1$ and $f_2$, minimize:

$$\sum_{\bar{f} \neq f_1} \max(0, \gamma - s_O(x, f_1) + s_O(x, \bar{f})) +$$

$$\sum_{\bar{f} \neq f_2} \max(0, \gamma - s_O([x, m_o_1], f_2) + s_O([x, m_o_1], \bar{f}])) +$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_o_1, m_o_2], r) + s_R([x, m_o_1, m_o_2], \bar{r})))$$

- Employing RNN for $R$ in MemNN: given $[x, o_1, o_2]$ to predict $r$
MemNN: Word Sequences as Input

Situation:

- Input: arriving in a word stream rather than sentence level
- Word sequences: not already segmented as statements and questions
MemNN: Word Sequences as Input

Situation:
- Input: arriving in a word stream rather than sentence level
- Word sequences: not already segmented as statements and questions

→ Add a segmentation function: sequences → sentences

\[ \text{seg}(c) = W_{\text{seg}}^T U S \Phi_{\text{seg}}(c) \]

where \( c \) is the input word sequence (BoW using a separate dictionary)
If \( \text{seg}(c) > \gamma \) (i.e. the margin), this sequence is recognised as a segment.

→ A learning component in MemNN’s write operation
MemNN: Efficient Memory via Hashing

Situation:

- The set of stored memories is very large
- Scoring all the memories to find the best supporting one is prohibitively expensive
MemNN: Efficient Memory via Hashing

Situation:

- The set of stored memories is very large
- Scoring all the memories to find the best supporting one is prohibitively expensive

→ Exploring hashing tricks to speed up lookup:
hash the input $I(x)$ into one or more buckets and then only score memories $m_i$ that are in the same buckets

- via hashing words: $|buckets| = |W|$
  For a given sentence: hash it into all the buckets corresponding to its words.
  A memory $m_i$ will only be considered if it shares at least one word with the input $I(x)$.

- via clustering word embeddings:
  For trained $U_O$, run $K$-means to cluster word vectors $(U_O)_i \rightarrow K$ buckets.
MemNN: Modeling Write Time

Answering questions about a story: relative order of events is important

→ Take in to account **when** a memory slot was written to

- Add extra features to $\Phi_x$ and $\Phi_y$ to encode absolute write time
- Learning a function on triples to get relative time order

\[
 s_{O_t}(x, y, y') = \Phi_x(x)^\top U_{O_t}^\top U_{O_t} \left( \Phi_y(y) - \Phi_y(y') + \Phi_t(x, y, y') \right)
\]

- extending the dimensionality of all the $\Phi$ embeddings by 3
- $\Phi_t(x, y, y')$ uses 3 new features (0-1 values):
  - whether $x$ is older than $y$, $x$ older than $y'$, and $y$ older than $y'$
- If $s_{O_t}(x, y, y') > 0$, the model prefers $y$; otherwise $y'$

→ choosing the best supporting memory: a loop over all the memories
- keeping the winning memory at each step
- always comparing the current winner to the next memory
Experiments: Large-Scale QA (Triple-KB)

Dataset:

- Pseudo-labeled QA pairs: (a question, an associated triple)
  - $14M$ statements (subject-relation-object triples):
    → stored as memories in the MemNN model
  - Triples: *REV ERB* extractions mined from the *ClueWeb09* corpus and cover diverse topics
  - Questions: generated from several seed patterns

- Paraphrased questions: $35M$ pairs from *WikiAnswers*

Task: re-ranking the top returned candidate answers by several systems measuring F1 score over the test set

MemNN Model: a $k = 1$ supporting memory with different variants

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Fader et al., 2013)</td>
<td>0.54</td>
</tr>
<tr>
<td>(Bordes et al., 2014b)</td>
<td>0.73</td>
</tr>
<tr>
<td>MemNN (embedding only)</td>
<td>0.72</td>
</tr>
<tr>
<td>MemNN (with BoW features)</td>
<td>0.82</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Embedding F1</th>
<th>Embedding + BoW F1</th>
<th>Candidates (speedup)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemNN (no hashing)</td>
<td>0.72</td>
<td>0.82</td>
<td>14M (0x)</td>
</tr>
<tr>
<td>MemNN (word hash)</td>
<td>0.63</td>
<td>0.68</td>
<td>13k (1000x)</td>
</tr>
<tr>
<td>MemNN (cluster hash)</td>
<td>0.71</td>
<td>0.80</td>
<td>177k (80x)</td>
</tr>
</tbody>
</table>
Experiments: Simulated World QA\(^5\)

Dataset:
- a simple simulation of 4 characters, 3 objects and 5 rooms
- characters: moving around, picking up and dropping objects
- statements (7\(k\) for training): generated text using a simple automated grammar based on actions
- questions (3\(k\) for training): mostly about people and position
- answers: single word answers OR a simple grammar for generating true answers in sentence form

→ a QA task on simple "stories"
- multiple statements have to be used to do inference
- the complexity of the task: controlled by setting a limit on the number of time steps in the past the entity we ask the question about was last mentioned
  - limit: 1, only the last mention
  - limit: 5, a random mention between 1-5 time steps in the past

\(^5\)http://fb.ai/babi
## Experiments: Simulated World QA (cont.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Difficulty 1</th>
<th>Difficulty 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>actor w/o before</td>
<td>actor</td>
</tr>
<tr>
<td>RNN</td>
<td>100%</td>
<td>60.9%</td>
</tr>
<tr>
<td>LSTM</td>
<td>100%</td>
<td>64.8%</td>
</tr>
<tr>
<td>MemNN $k = 1$</td>
<td>97.8%</td>
<td>31.0%</td>
</tr>
<tr>
<td>MemNN $k = 1$ (+time)</td>
<td>99.9%</td>
<td>60.2%</td>
</tr>
<tr>
<td>MemNN $k = 2$ (+time)</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Figure 2: Sample test set predictions (in red) for the simulation in the setting of word-based input and where answers are sentences and an LSTM is used as the $R$ component of the MemNN.

Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living room.
Where is Dan? **A: living room I believe**
Where is Joe? **A: the bathroom**
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now? **A: the milk is in the kitchen**
Where is Dan now? **A: I think he is in the bedroom**
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? **A: I think Joe is in the office**
Combined Experiments

Combining simulated world learning with real-world data:

- to show the power and generality of the MemNN models
- build an ensemble of MemNN models trained on large-scale QA and simulated data
- to answer both general knowledge questions and specific statements relating to the previous dialogue

Figure 4: An example dialogue with a MemNN system trained on both the simulation data and the large-scale QA data. The system is able to (attempt to) answer questions about general world knowledge and about specific story-based statements in the dialogue.
Outline

Introduction

MemNN: Memory Networks

MemNN-WSH: Weakly Supervised Memory Networks
  Introduction
  MemNN-WSH: Memory via Multiple Layers
  Experiments
Introduction

MemNN: Strongly Supervised Memory Networks

- Explore how explicit long-term storage can be combined with neural networks
- Need extensive supervision to train:
  - The ground truth answer
  - Explicit indication of the supporting sentences within the text

MemNN-WSH: Weakly Supervised Memory Networks

- Learn with weak supervision:
  just the answer, without the need for support labels
- Enable the model to operate in more general settings where carefully curated training data is not available
- Demonstrate that a long-term memory can be integrated into neural network models that rely on standard input/output pairs for training

→ A content-based memory system:

- Using continuous functions for the read operation
- Sequentially writing all inputs up to a fixed buffer size
Task Introduction

- A given bAbl\(^6\) task consists of a set of statements, followed by a question whose answer is typically a single word (in a few tasks, answers are a set of words).
- There are a total of 20 different types of bAbl tasks that probe different forms of reasoning and deduction.
- Formal Task Description:
  - For one of the 20 bAbl tasks, we are given \(P\) example problems, each having a set of \(I\) sentences \(x_i^p\) where \(I\leq 20\); a question sentence \(q^p\) and answer \(a^p\).
  - The examples are randomly split into disjoint train and test sets
  - Let the \(j\)th word of sentence \(i\) be \(x_{ij}\), represented by a one-hot vector of length \(V\) (where \(|V| = 177\) since the bAbl language is very simple).

\(^6\)http://fb.ai/babi
MemNN-WSH: Single Layer for a single memory lookup operation

INPUT Side:
implementing content-based addressing, with each memory location holding a distinct output vector

• For the memory:
  Given an input sentence (a statement of facts): \( x_i = \{x_{i1}, x_{i2}, ..., x_{in}\} \)
  The memory vector \( m_i \in \mathbb{R}^d: m_i = \sum_j A x_{ij} \)

• For the question:
  The question vector \( q \) is also embedded via matrix \( B \): \( u = \sum_j B q_j \)

• For the match between the question \( u \) and each memory \( m_i \):
  The probability vector: \( p_i = \text{softmax}(u^T m_i) = \text{softmax}(q^T B^T \sum_j A x_{ij}) \)

OUTPUT Side:

• Each memory vector on the input has a corresponding output vector \( c_i \):
  \( c_i = \sum_j C x_{ij} \)

• The output vector \( o \) from the memory:
  \( o = \sum_i p_i c_i = \sum_i \sum_j p_i C x_{ij} \)
ANSWER Prediction:

- The sum of the output vector $o$ and the question embedding $u$ is passed through a final weight matrix $W$ to produce the answer $\hat{a}$:
  \[
  \hat{a} = \text{softmax}(W(o + u))
  \]

Parameters $A, B, C$ and $W$ are jointly learned by minimizing a standard cross-entropy loss between $\hat{a}$ and the true answer $a$. 

![Diagram of MemNN-WSH: Single Layer](image_url)
MemNN-WSH: Multiple Layers

The single memory layer: only able to answer questions that involve a single memory lookup.

*If a retrieved memory depends on another memory*, then multiple lookups are required to answer the question.

The memory layers are stacked in the following way:

- Input of \((k+1)\)th layer is the sum of the output \(o^k\) and the input \(u^k\) from layer \(k\):
  \[ u^{k+1} = u^k + o^k \]

- Each layer has its own embedding matrices \(A^k, C^k\)
  
  - Adjacent: the output embedding for one layer is the input embedding for the one above:
    \[ A^{k+1} = C^k \]
  
  - Layer-wise (RNN): the input and output embeddings are the same across different layers:
    \[ A^1 = A^2 = A^3, C^1 = C^2 = C^3 \]

- At the top of the network, the answer is predicted as:
  \[ \hat{a} = \text{softmax}(W(o^K + u^K)) \]
MemNN-WSH: Multiple Layers (cont.)
MemNN-WSH

Sentence Representation:
- BoW representation: the sum of words $m_i = \sum_j A x_{ij}$
- PE representation: Encoding the position of words within the sentence (used for questions, memory inputs and memory outputs)
  $$m_i = \sum_j l_j \cdot A x_{ij}$$
  where $\cdot$ an element-wise multiplication
  $l_j$ is a column vector with the structure
  $$l_{kj} = (1 - j/J) - (k/d) (1 - 2j/J)$$
  $J$ is the number of words in the sentence
  $d$ is the dimension of the embedding

Temporal Encoding: Relative order of events
- Add notion of temporal context: $m_i = \sum_j A x_{ij} + T_A(i)$
- Augment the output in the same way: $c_i = \sum_j C x_{ij} + T_C(i)$
- Learning time invariance by injecting random noise: add dummy memories to regularize temporal parameters
Experiments

Settings:
- The bAbI QA dataset (2 versions): 1k and 10k training problems per task
- All experiments: a 3 layer model - 3 memory lookups
- Weight sharing scheme: Adjacent
- Output lists: take each possible combination of possible outputs and record them as a separate answer vocabulary word

Baselines:
- **MemNN**: the strongly supervised Memory Networks (using best reported approach in the previous paper)
- **MemNN-WSH**: a weakly supervised heuristic version of MemNN
  - the first hop memory should share at least one word with the question
  - the second hop memory should share at least one word with the first hop and at least one word with the answer
  - All those memories that conform are called **valid memories**
  - The training objective: learning a margin ranking loss function to rank valid memories higher than invalid memories
- **LSTM**: a standard LSTM model trained only with QA pairs
Experiments (cont.)

Exploring a variety of design choices:

- BoW vs Position Encoding (PE) sentence representation
- training on all 20 tasks jointly \((d=50)\) vs independent training \((d=20)\)

<table>
<thead>
<tr>
<th>Task</th>
<th>Supervised</th>
<th>Weakly Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>Ours</td>
</tr>
<tr>
<td></td>
<td>MemNN</td>
<td>PE</td>
</tr>
<tr>
<td>1: 1 supporting fact</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2: 2 supporting facts</td>
<td>0.0</td>
<td>11.0</td>
</tr>
<tr>
<td>3: 3 supporting facts</td>
<td>0.0</td>
<td>14.7</td>
</tr>
<tr>
<td>4: 2 argument relations</td>
<td>0.0</td>
<td>21.8</td>
</tr>
<tr>
<td>5: 3 argument relations</td>
<td>2.0</td>
<td>5.4</td>
</tr>
<tr>
<td>6: yes/no questions</td>
<td>0.0</td>
<td>12.6</td>
</tr>
<tr>
<td>7: counting</td>
<td>15.0</td>
<td>20.2</td>
</tr>
<tr>
<td>8: lists/sets</td>
<td>9.0</td>
<td>18.5</td>
</tr>
<tr>
<td>9: simple negation</td>
<td>0.0</td>
<td>16.2</td>
</tr>
<tr>
<td>10: indefinite knowledge</td>
<td>2.0</td>
<td>34.8</td>
</tr>
<tr>
<td>11: basic coherence</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>12: conjunction</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>13: compound coherence</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>14: time reasoning</td>
<td>1.0</td>
<td>0.2</td>
</tr>
<tr>
<td>15: basic deduction</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>16: basic induction</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>17: positional reasoning</td>
<td>35.0</td>
<td>54.9</td>
</tr>
<tr>
<td>18: size reasoning</td>
<td>5.0</td>
<td>38.0</td>
</tr>
<tr>
<td>19: path finding</td>
<td>64.0</td>
<td>83.2</td>
</tr>
<tr>
<td>20: agent’s motivation</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Mean error (%)</td>
<td>6.7</td>
<td>16.6</td>
</tr>
<tr>
<td>Failed tasks (err. &gt; 5%)</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 1: Test error rates (%) on the 20 bAbI tasks for models using 1k training examples. Key: BoW = bag-of-words representation; PE = position encoding representation; LS = linear start training; RN = random injection of time index noise; RNN = RNN-style layer-wise weight tying (if not stated, adjacent weight tying is used); joint = joint training on all tasks (as opposed to per-task training).
### Experiments (cont.)

<table>
<thead>
<tr>
<th>Task</th>
<th>Supervised</th>
<th>Weakly Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>Ours</td>
</tr>
<tr>
<td></td>
<td>MemNN</td>
<td>MemNN-WSH</td>
</tr>
<tr>
<td>1: 1 supporting fact</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>2: 2 supporting facts</td>
<td>0.0</td>
<td>39.6</td>
</tr>
<tr>
<td>3: 3 supporting facts</td>
<td>0.0</td>
<td>79.5</td>
</tr>
<tr>
<td>4: 2 argument relations</td>
<td>0.0</td>
<td>36.6</td>
</tr>
<tr>
<td>5: 3 argument relations</td>
<td>0.3</td>
<td>21.1</td>
</tr>
<tr>
<td>6: yes/no questions</td>
<td>0.0</td>
<td>49.9</td>
</tr>
<tr>
<td>7: counting</td>
<td>3.3</td>
<td>35.1</td>
</tr>
<tr>
<td>8: lists/sets</td>
<td>1.0</td>
<td>42.7</td>
</tr>
<tr>
<td>9: simple negation</td>
<td>0.0</td>
<td>36.4</td>
</tr>
<tr>
<td>10: indefinite knowledge</td>
<td>0.0</td>
<td>76.0</td>
</tr>
<tr>
<td>11: basic coherence</td>
<td>0.0</td>
<td>25.3</td>
</tr>
<tr>
<td>12: conjunction</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>13: compound coherence</td>
<td>0.0</td>
<td>12.3</td>
</tr>
<tr>
<td>14: time reasoning</td>
<td>0.0</td>
<td>8.7</td>
</tr>
<tr>
<td>15: basic deduction</td>
<td>0.0</td>
<td>68.8</td>
</tr>
<tr>
<td>16: basic induction</td>
<td>0.0</td>
<td>50.9</td>
</tr>
<tr>
<td>17: positional reasoning</td>
<td>24.6</td>
<td>51.1</td>
</tr>
<tr>
<td>18: size reasoning</td>
<td>2.1</td>
<td>45.8</td>
</tr>
<tr>
<td>19: path finding</td>
<td>31.9</td>
<td>100.0</td>
</tr>
<tr>
<td>20: agent's motivation</td>
<td>0.0</td>
<td>4.1</td>
</tr>
<tr>
<td><strong>Mean error (%)</strong></td>
<td>3.2</td>
<td>39.2</td>
</tr>
<tr>
<td><strong>Failed tasks (err. &gt; 5%)</strong></td>
<td>2</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 2: Test error rates (%) on the 20 bAbI tasks for models using 10k training examples. Key: BoW = bag-of-words representation; PE = position encoding representation; LS = linear start training; RNN = RNN-style layer-wise weight tying (if not stated, adjacent weight tying is used); joint = joint training on all tasks (as opposed to per-task training).
Outline

Introduction

MemNN: Memory Networks
  Memory Networks: General Framework
  MemNNs for Text
  Experiments

MemNN-WSH: Weakly Supervised Memory Networks
  Introduction
  MemNN-WSH: Memory via Multiple Layers
  Experiments
Appendix: Gated Recurrent Neural Networks

LSTM & GRU


LSTM Unit:

\[
\begin{align*}
    h_t^j &= o_t^j \tanh(c_t^j) \\
    o_t^j &= \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^j \\
    c_t^j &= f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \\
    \tilde{c}_t^j &= \tanh(W_c x_t + U_c h_{t-1})^j \\
    f_t^j &= \sigma(W_f x_t + U_f h_{t-1} + V_f c_t)^j \\
    i_t^j &= \sigma(W_i x_t + U_i h_{t-1} + V_i c_t)^j
\end{align*}
\]

GRU Unit:

\[
\begin{align*}
    h_t^j &= (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j \\
    z_t^j &= \sigma(W_z x_t + U_z h_{t-1})^j \\
    \tilde{h}_t^j &= \tanh(Wx_t + Ur h_{t-1})^j \\
    r_t^j &= \sigma(Wr x_t + Ur h_{t-1})^j
\end{align*}
\]