Neural Turing Machines & Dynamic Memory Networks

Dr. Kira Radinsky
CTO SalesPredict
Visiting Professor/Scientist Technion

Slides were adapted from lecture by Richard Socher, Alex Graves, Greg Wayne & Ivo Danihelka
Neural Turing Machines

Can neural nets learn programs?
Learning to Execute - try #1

Can LSTM learn to execute python code?

**Input:**

```python
j=8584
for x in range(8):
    j+=920
b=(1500+j)
print((b+7567))
```

**Target:** 25011.

**Input:**

```python
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```

**Target:** 12184.

LSTM reads the entire input one character at a time and produces the output one character at a time.
Learning to Execute - try #1

**Training data:** Python short code that can be evaluated in O(n) time & O(1) memory, with the following characteristics:

- **operators:**
  - addition, subtraction, multiplication, variable assignments, if statements, and for loops, but not double loops.

- **length parameter:**
  - constrain the integer in a maximum length.

- **nesting parameter:**
  - constrain the number of times to combine operations.

**Input:**

```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```

**Target:** 12184.

an example of length = 4, nesting = 3
curriculum learning

A trick for learning that gradually increase the difficulties of training examples.

baseline: training examples with length = a, nesting = b.
naive: start with length = 1, nesting = 1 and gradually increase until length = a, nesting = b.
mix: to generate a example, first pick a random length from [1, a], and a random nesting from [1, b].
combined: a combination of naive and mix.
Evaluation

Use teacher forcing when predicting the i-th digit of the target, the LSTM is provided with the correct first i-1 digits.

Results

Torch code available: https://github.com/wojciechz/learning_to_execute
Neural Turing Machines – Background

• First application of Machine Learning to logical flow and external memory
• Extend the capabilities of neural networks by coupling them to external memory
• Analogous to TM coupling a finite state machine to infinite tape
• RNN’s have been shown to be Turing—Complete, Siegelmann et al ‘95
• Unlike TM, NTM is completely differentiable
Neural Turing Machines - Foundational Research

• Neuroscience and Psychology
  – Concept of “working memory”: short-term memory storage and rule based manipulation
  – Also known as “rapidly created variables”
  – Observational neuroscience results in the prefrontal cortex and basal ganglia of monkeys

• Cognitive Science and Linguistics
  – AI and Cognitive Science were contemporaneous in 1950’s-1970’s
  – Two fields parted ways when neural nets received criticism, Fodor et al. ’88
  – Motivated Recurrent Networks research to handle variable binding and variable length input
  – Recursive processing hot debate topic in role inhuman evolution (Pinker vs Chomsky)

• Recurrent Neural Networks
• 5 days after the memory networks…
Neural Turing Machines

In QA memory network, memory is mainly used for a knowledge database. Interaction between computation resources and memory is very limited.

Neural-Turing machine proposes an addressing mechanism as well as coupled reading & writing operations.
Neural Turing Machines
Neural Turing Machines
Neural Turing Machines

Let $M_t$ be the memory matrix of size $NxM$,
- $N$ is the number of memory locations
- $M$ is the vector size at each location.

in order for this to be able to be “end-to-end” differentiable

Read:
$$\sum_i w_t(i) = 1, \quad 0 \leq w_t(i) \leq 1$$
$$r_t \leftarrow \sum_i w_t(i) M_t(i)$$

Write:
erase:
$$\tilde{M}_t(i) \leftarrow M_{t-1}(i)[1 - w_t(i)e_t]$$
add:
$$M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t$$
Neural Turing Machines

**Intuition**: \(w_t\) is used as a focusing mechanism to select the row, and \(r_t\), \(e_t\) and \(a_t\) perform their individual roles element wise in \(M_t(i)\).

\[
M_t = \begin{bmatrix}
M_t(0)_0 & \cdots & M_t(0)_{N-1} \\
\vdots & \ddots & \vdots \\
M_t(i)_0 & \cdots & M_t(i)_{N-1} \\
\vdots & \ddots & \vdots \\
M_t(M-1)_0 & \cdots & M_t(M-1)_{N-1}
\end{bmatrix}
\]

\[
w_t = \begin{bmatrix}
w_t(0) \\
\vdots \\
w_t(i) \\
w_t(M-1)
\end{bmatrix}
\]

\[
r_t = \begin{bmatrix}
r_0 & \cdots & r_{N-1}
\end{bmatrix}
\]

\[
e_t = \begin{bmatrix}
e_0 & \cdots & e_{N-1}
\end{bmatrix}
\]

\[
a_t = \begin{bmatrix}
a_0 & \cdots & a_{N-1}
\end{bmatrix}
\]
Neural Turing Machines

1. Reading
2. Writing (involves both erasing and adding)
3. Addressing
Neural Turing Machines

1. Reading
2. Writing (involves both erasing and adding)
3. Addressing
   1. **Find Memory Key**: Take in an input from the input sequence, and translate it to some sort of key ($K_t$) in the memory
   2. **Shift**: merely picking out the value at $K_t$ may not be particularly useful, since we could possibly have just “used” the prediction.
      - We might need to shift ($S_t$) before or after that memory location to find something useful for computation
Neural Turing Machines

3. Addressing - **Focusing by Content**
   - Each head (write\read – the controller decides which) produces key vector $k_t$ of length $M$
   - We look up at $M_t$ to find the entry most similar to $k_t$ – but probabilistically!
     - Done by a similarity function (in this case cosine similarity computed between $k_t$ over all entries in $M_t$)
     - Generate a content based weight $w_t^c$ based on the similarity measure, using ‘key strength’ $\beta_t$

\[
    w_t^c(i) \leftarrow \frac{\exp \left( \beta_t K[k_t, M_t(i)] \right)}{\sum_j \exp \left( \beta_t K[k_t, M_t(j)] \right)}.
\]

\[
    K[u, v] = \frac{u \cdot v}{||u|| \cdot ||v||}.
\]

just an intermediate value before the final weight vector is computed.
Neural Turing Machines

3. Addressing – how to shift? (interpolation)
   - Use $w_t^c$ (weighting for content-based addressing) or perform a shift from the weighting from the previous time step?
   - Each head emits a scalar interpolation gate $g_t$

\[
    w_t^g \leftarrow g_t w_t^c + (1 - g_t) w_{t-1}.
\]

the “expected” value $w_{t-1}$ and $w_t^c$
Neural Turing Machines

3. Addressing – convolutional shift

- Shift $w_t^g$ probabilistically.
- Each head emits a distribution over allowable integer shifts $s_t$

$$
\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s_t(i - j) \quad \sum_i s_t(i) = 1
$$

In vectorized form:

$$
S_t = \begin{bmatrix}
    s_t(0) & s_t(N - 1) & \cdots & s_t(2) & s_t(1) \\
    s_t(1) & s_t(0) & s_t(N - 1) & \cdots & s_t(2) \\
    \vdots & s_t(1) & s_t(0) & \ddots & s_t(2) \\
    s_t(N - 2) & \ddots & \ddots & s_t(N - 1) \\
    s_t(N - 1) & s_t(N - 2) & \cdots & s_t(1) & s_t(0)
\end{bmatrix}
$$

$$
\tilde{w}_t = S_t w_t^g
$$
3. Addressing – sharpening

- Each head emits a scalar sharpening parameter $\gamma_t$

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i) \gamma_t}{\sum_j \tilde{w}_t(j) \gamma_t}$$
3. Addressing (putting it all together)

This can operate in three complementary modes

- A weighting can be chosen by the content system without any modification by the location system
- A weighting produced by the content addressing system can be chosen and then shifted
- A weighting from the previous time step can be rotated without any input from the content-based addressing system
Neural Turing Machines – the controller

• Think of it as the CPU of the entire system, deciding what to read and write to memory.
• The authors experiment with using a LSTM neural network and a standard feed-forward network for this purpose.

**Input** (fed into the controller):

\[ i_t, r_t \in \mathbb{R}^N \]

**Output with which we use to manipulate \( M_t \):**

\[ e_t, a_t, k_t \in (0, 1)^N \]
\[ s_t \in (0, 1)^M, \quad \sum_i s_t(i) = 1 \]
\[ \beta_t \in \mathbb{R}^+ \]
\[ \gamma_t \in \mathbb{R}^{\geq 1} \]
\[ g_t \in (0, 1) \]
Experiments

• Test NTM’s ability to learn simple algorithms like copying and sorting
• Demonstrate that solutions generalize well beyond the range of training
• Tests three architectures
  – NTM with feed forward controller
  – NTM with LSTM controller
  – Standard LSTM network
Experiments - Copy

NTM is presented with an input sequence of random binary vectors, and asked to recall it.

- Tests whether NTM can store and retrieve data
- Trained to copy sequences of 8 bit vectors
- Sequences vary between 1-20 vectors
Experiments - Copy
Experiments - Copy

Sequences of length 10

Sequences of length 20

Sequences of length 30

Sequences of length 50

- NTM

Sequences of length 120

- LSTM
Intermediate variables suggest the following copy algorithm.

**initialise:** move head to start location
**while** input delimiter not seen **do**
  receive input vector
  write input to head location
  increment head location by 1
**end while**
return head to start location
**while** true **do**
  read output vector from head location
  emit output
  increment head location by 1
**end while**
Repeat Copy: NTM is presented with an input sequence and a scalar indicating the number of copies.

- Tests whether NTM can learn simple nested “for loop” function
- Extend copy by repeatedly copying input specified number of times
- Training is a random-length sequence of 8 bit binary inputs plus a scalar value for # of copies
- Scalar value is random between 1-10
Figure 7: Repeat Copy Learning Curves.
Experiments - Repeat Copy

**NTM**
Length 10, Repeat 20

<table>
<thead>
<tr>
<th>Targets</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Length 20, Repeat 10

<table>
<thead>
<tr>
<th>Targets</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LSTM**
Length 10, Repeat 20

<table>
<thead>
<tr>
<th>Targets</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Length 20, Repeat 10

<table>
<thead>
<tr>
<th>Targets</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fails to figure out where to end. Unable to keep count of how many repeats it has completed.

• Use another memory location to help switch back the pointer to the start.
Experiments - Associative Recall

NTM is presented with a sequence and a query, then it is asked to output datum behind the query.

- To test if NTM can apply algorithms to relatively simple, linear data structures.
- Tests NTM’s ability to associate data references
- Training input is list of items, followed by a query item
- Output is subsequent item in list
- Each item is a three sequence 6-bit binary vector
- Each ‘episode’ has between two and six items
Experiments - Associative Recall
Experiments - Associative Recall

Figure 11: Generalisation Performance on Associative Recall for Longer Item Sequences. The NTM with either a feedforward or LSTM controller generalises to much longer sequences of items than the LSTM alone. In particular, the NTM with a feedforward controller is nearly perfect for item sequences of twice the length of sequences in its training set.
Experiments - Associative Recall

• when each item delimiter is presented, the controller writes a compressed representation of the previous three time slices of the item.

• After the query arrives, the controller recomputes the same compressed representation of the query item, uses a content-based lookup to find the location where it wrote the first representation, and then shifts by one to produce the subsequent item in the sequence.
Experiments - Priority Sort

A sequence of random binary vectors is input to the network along with a scalar priority rating for each vector

- Tests whether NTM can sort data
- Input is sequence of 20 random binary vectors, each with a scalar rating drawn from $[-1, 1]$
- Target sequence is 16-highest priority vectors
Experiments - Priority Sort

Hypothesis that NTM uses the priorities to determine the relative location of each write. The network reads from the memory location in an increasing order.
Experiments - Priority Sort

![Graph showing cost per sequence (bits) vs. sequence number (thousands)]
Dynamic Memory Networks

Can all NLP tasks be reduced to question answering?
• Question answering tackles complex questions over lots of text
  • Where was Obama's wife born?
• Machine translation
  • What is the translation into French?
• Sequence modeling tasks like named entity recognition (NER)
  • What are the named entity tags in this sentence?
• Classification problems like sentiment analysis
  • What is the sentiment?
• Even multi-sentence joint classification problems like coreference resolution
  • Who does "their" refer to?
Reduction to QA

Interesting but useless?
Ask Me Anything: Dynamic Memory Networks for NLP
Example Input, Question, Answer

I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden

I: Everybody is happy.
Q: What’s the sentiment?
A: positive

I: Jane has a baby in Dresden.
Q: What are the named entities?
A: Jane - person, Dresden - location

I: Jane has a baby in Dresden.
Q: What are the POS tags?
A: NNP VBZ DT NN IN NNP .

I: I think this model is incredible
Q: In French?
A: Je pense que ce modèle est incroyable.
The DMN

Semantic Memory
- Word vectors
- Knowledge Basis

Episodic Memory

Input Text Sequence

Question

Answer
The Modules: Input

- Responsible for computing representations of anything of type sequence: (audio, visual or) textual inputs such that they can be retrieved when needed later.
- Assume a temporal sequence indexable by a time stamp.
- For written language we have a sequence of words \((v_1,...,v_{Tw})\)
- Context-independent (e.g. word vectors) and context-dependent hidden states
- Word vectors from Glove model Pennington et al. (2014) → Stored in semantic memory module
- RNN computation for context states →
Reminder: Gated Recurrent Units in RNN

\[
h_t = GRU(x_t, h_{t-1})
\]

\[
\begin{align*}
z_t &= \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} + b^{(z)} \right) \\
r_t &= \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} + b^{(r)} \right) \\
\tilde{h}_t &= \tanh \left( W x_t + r_t \circ U h_{t-1} + b^{(h)} \right) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t,
\end{align*}
\]

For DMN input sequence: \( w_t = GRU(v_t, w_t) \).
The Modules: Question

- Simple GRU over question word vectors: \( q_t = GRU(v_t, q_{t-1}) \)

The input module is a gated recurrent unit (GRU) running on the sequence of word vectors (output after sentences or works). The question model uses the same GRU (processed word by word and outputs one vector at the end.)
The Modules: Episodic Memory

• Combines the previous three modules' (semantic, input and query) outputs in order to reason over them and give the resulting knowledge to the answer module.
• Dynamically retrieves the necessary information over the sequence of words or sentences.
• If necessary to retrieve additional facts \(\rightarrow\) repeat over inputs
• Needed for transitive inference (TI)
  • The hippocampus, the seat of episodic memory in humans, is active during this kind of inference and disruption of the hippocampus impairs TI
The Modules: Episodic Memory

- The fact and question vectors extracted from the input enter the episodic memory module.
- The episodic memory is a composition of two nested GRUs.
  - **The outer GRU** generates the **final memory vector** working over a sequence of so-called episodes.
    - initialized by the question vector.
  - **The inner GRU** generates the **episodes** by passing over the facts from the input module.
    - When updating its inner state takes into account the output of some attention function on the current fact (0-1)
    - Takes into account the output of some attention function on the current fact.
      - Attention function is a simple 2-layer neural network depending on the question vector, current fact, and current state of the memory.
The Modules: Episodic Memory

- After each full pass on all facts the inner GRU outputs an *episode* which is fed into the outer GRU which on its turn updates the memory.
  - Then because of the updated memory the attention may give different scores to the facts and new episodes can be created.
- The number of steps of the outer GRU, that is a parameter of the system.
- All facts, episodes and memories are in the same n-dimensional space.
Gates over input sentences

• For each sentence in input:

\[ z(s, m, q) = [s \circ q, s \circ m, |s - q|, |s - m|, s, m, q, s^T W^{(b)} q, s^T W^{(b)} m] \]

Scoring function: \( G(s, m, q) = \sigma \left( W^{(2)} \tanh \left( W^{(1)} z(s, m, q) + b^{(1)} \right) + b^{(2)} \right) \)

• Summarize important facts in episode vector:

\[ e^1 = \sum_{t=1}^{T} \text{softmax}(g^1_t) s_t \]

• Done if only one pass over data was needed to answer question
Episodes

• What about: (from Facebook babI dataset)

  I: Mary walked to the bathroom.
  I: Sandra went to the garden.
  I: Daniel went back to the garden.
  I: Sandra took the milk there.
  Q: Where is the milk?
  A: garden
Episodes

• Iterate over multiple episodes

• Compute new gates (second episode) with previous memory vector:

\[ g_t^2 = G(s_t, m^1, q) \]

• GRU over memories:

\[ m^1 = GRU(e^1, m^0) \]
The Modules: Answer

- Simple GRU to produce an output at each of its time steps.
- Allow to predict EOS token and stop

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

* we concatenate the last generated word and the question vector as the input at each time step.
Putting it all together

- Training via cross-entropy errors and backpropagation
<table>
<thead>
<tr>
<th>Type</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA</td>
<td>babI - Facebook</td>
</tr>
<tr>
<td>Sequence</td>
<td>POS</td>
</tr>
<tr>
<td>Classification</td>
<td>Sentiment</td>
</tr>
<tr>
<td>Sequence</td>
<td>NER</td>
</tr>
<tr>
<td>MT</td>
<td>English-French</td>
</tr>
<tr>
<td>Coref</td>
<td>Guha et al. 2015</td>
</tr>
</tbody>
</table>
## Details: QA on babI, POS and Sentiment

<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>99.3</td>
<td>14: Time Reasoning</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>5: Three Argument Relations</td>
<td>98</td>
<td>97.5</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><strong>Mean Accuracy (%)</strong></td>
<td></td>
<td></td>
<td></td>
<td>93.3</td>
<td>93.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>SVMTool</th>
<th>Sogaard</th>
<th>Suzuki et al.</th>
<th>Spoustova et al.</th>
<th>SCNN</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc (%)</td>
<td>97.15</td>
<td>97.27</td>
<td>97.40</td>
<td>97.44</td>
<td>97.50</td>
<td></td>
</tr>
<tr>
<td><strong>Acc (%)</strong></td>
<td><a href="#">97.56</a></td>
<td><a href="#">97.56</a></td>
<td><a href="#">97.56</a></td>
<td><a href="#">97.56</a></td>
<td><a href="#">97.56</a></td>
<td><a href="#">97.56</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>MV-RNN</th>
<th>RNTN</th>
<th>DCNN</th>
<th>PVec</th>
<th>CNN-MC</th>
<th>DRNN</th>
<th>CT-LSTM</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>82.9</td>
<td>85.4</td>
<td>86.8</td>
<td>87.8</td>
<td>88.1</td>
<td>86.6</td>
<td>88.0</td>
<td>88.3</td>
</tr>
<tr>
<td>Fine-grained</td>
<td>44.4</td>
<td>45.7</td>
<td>48.5</td>
<td>48.7</td>
<td>47.4</td>
<td>49.8</td>
<td>51.0</td>
<td>50.3</td>
</tr>
</tbody>
</table>
Dynamic Memory Network by MetaMind

Story
wolves are afraid of mice.
sheep are afraid of mice.
winona is a sheep.
mice are afraid of cats.
cats are afraid of wolves.
jessica is a mouse.
emily is a cat.
gertrude is a wolf.

Question
what is winona afraid of?

Answer: mouse

Episode 1
0.00: wolves are afraid of mice
0.00: sheep are afraid of mice
0.99: winona is a sheep
0.00: mice are afraid of cats
0.00: cats are afraid of wolves
0.00: jessica is a mouse
0.00: emily is a cat
0.01: gertrude is a wolf

Episode 2
0.00: wolves are afraid of mice
1.00: sheep are afraid of mice
0.00: winona is a sheep
0.00: mice are afraid of cats
0.00: cats are afraid of wolves
0.00: jessica is a mouse
0.00: emily is a cat
0.00: gertrude is a wolf

Model hidden state
Summary

- Introduced an neural net architecture with external memory that is differentiable end-to-end
- Experiments demonstrate that NTM are capable of leaning simple algorithms and are capable of generalizing beyond training regime
- All (?) NLP tasks can be reduced to question answering
- The DMN can very accurately train with input-question-answer triplets
- Next steps: One very large multitask DMN?
“Again, it [the Analytical Engine] might act upon other things besides numbers… the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.” — Ada Lovelace