Attention and Memory Networks – for translation, question answering and others

Dr. Kira Radinsky
CTO SalesPredict
Visiting Professor/Scientist Technion

Slides were adapted from lecture by Jason Weston (Facebook)
RNN Encoder-Decoder Framework

Cho et al. (2014a) and Sutskever et al. (2014) architecture that learns to align and translate simultaneously.
An encoder reads the input sentence (sequence of vectors) into a vector \( c \) (the context).

The most common approach is to use an RNN.

E.g. Sutskever et al. (2014) used an LSTM as \( f \) and \( q(\{h_1, \ldots, h_T\}) = h_T \), for instance.
Decoder

- The decoder is often trained to predict the next word $y_t$ given the context vector $c$ and all the previously predicted words $\{y_1, \cdots, y_{t-1}\}$

- Defines a probability over the translation $y$ by decomposing the joint probability into the ordered conditionals

$$p(y) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c),$$

- With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$$

nonlinear, potentially multi-layered, function that outputs the probability of $y_t$

the hidden state of the RNN
Attention

Neural machine translation by jointly learning to align and translate (Bahdanau, Cho and Bengio)

Each conditional probability

$ p(y_i | y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, s_i, c_i) $,

RNN hidden state for time $ i $:

$s_i = f(s_{i-1}, y_{i-1}, c_i)$

The context vector based on the encoder annotations $ h_i $:

$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$

Weight of each annotation $ h_j $:

$ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} $

alignment model which scores how well the inputs around position $ j $ and the output at position $ i $ match:

$ e_{ij} = a(s_{i-1}, h_j) $,

just a single-layer multilayer perceptron:

$ a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j) $,

The graphical illustration of the proposed model trying to generate the t-th target word $ y_t $ given a source sentence $ (x_1, x_2, \ldots, x_T) $.

The alignment model $ a $ is parameterized as a feed-forward neural network which is jointly trained with all the other components of the proposed system.
Attention - Visualization

For example, by visualizing the attention weight matrix when a sentence is translated, we can understand how the model is translating:

Each pixel shows the weight $\alpha_{ij}$ of the annotation of the j-th source word for the i-th target word.
Cost of Attention

- We need to calculate an attention value for each combination of input and output word
  - 50-word input sequence and generate a 50-word output sequence that would be 2500 attention values
- The model outputs a translated word, and then goes back through all of the internal memory of the text in order to decide which word to produce next.
- Counterintuitive.... human attention is something that’s supposed to save computational resources.
  - By focusing on one thing, we can neglect many other things
- Solution: use Reinforcement Learning to predict an approximate location to focus to.
Consider the attention problem as the sequential decision process of a goal-directed agent interacting with a visual environment. At each point in time, the agent observes the environment with a bandwidth-limited sensor (attention). Agent can actively control the sensor location. Acts based on the partial info. Agent learns over time how to act and how to deploy its sensor most effectively. At each step, the agent receives a scalar reward based on actions. Agent's Goal: maximize the total sum of such rewards.
RL for Attention
Recurrent Models of Visual Attention Mnih, Heess, Graves & Kavukcuoglu

• **Sensor**: At each step \( t \) the agent receives a (partial) observation of the environment in the form of an image \( x_t \).

• **Internal state**: the hidden units \( h_t \) of the recurrent neural network

• **Actions**:  
  • decides how to deploy its sensor (e.g. stochastically from Gaussian distribution depending on \( h_t \))  
  • an environment action at which might affect the state of the environment  

\[
a_t \sim p(\cdot | f_a(h_t; \theta_a))
\]

(e.g., in classification, modeled by softmax)

• **Reward**:  
  \[
  R = \sum_{t=1}^{T} r_t
  \]
  • In the case of object recognition, \( r_T = 1 \) if the object is classified correctly after \( T \) steps and 0 otherwise.
RL for Attention
Recurrent Models of Visual Attention Mnih, Heess, Graves & Kavukcuoglu

• Maximize the reward

\[ J(\theta) = \mathbb{E}_{p(s_{1:T}; \theta)} \left[ \sum_{t=1}^{T} r_t \right] = \mathbb{E}_{p(s_{1:T}; \theta)} [R]. \]

• Viewing the problem as a POMDP, allows to bring techniques from the RL literature

\[ \nabla_{\theta} J = \sum_{t=1}^{T} \mathbb{E}_{p(s_{1:T}; \theta)} [\nabla_{\theta} \log \pi(u_t|s_{1:t}; \theta) R] \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_{\theta} \log \pi(u_t^i|s_{1:t}^i; \theta) R^i, \]

where s i’s are interaction sequences obtained by running the current agent i, for i = 1 . . . M episodes.
Attention for Generating Image Descriptions

- Convolutional Neural Network to “encode” the image
- Recurrent Neural Network with attention mechanisms to generate a description

Neural Image Caption Generation with Visual Attention (Xu et al. 2015)
Attention for Generating Image Descriptions
Attention for Generating sentence parse trees

Recurrent Neural Network with attention mechanism to generate sentence parse trees

Grammar as a Foreign Language (Vinyals et al. 2014)
Attention for Question Answering

RNN to read a text, read a (synthetically generated) question, and then produce an answer

Teaching machines to read and comprehend (Hermann et al. 2015)
From Attention to Memory

- Attention mechanism allows the network:
  - to refer back to the input sequence, instead of forcing it to encode all information into one fixed-length vector
  - access to its internal memory (the hidden state of the encoder)
- “The network chooses what to retrieve from memory.”
- The network retrieves a weighted combination of all memory locations, not a value from a single discrete location
From Attention to Memory

• The trend today is towards memory networks
  • allow the network to read same input sequence **multiple times** before making an output, updating the memory contents at each step

• LSTM is kind of a memory
  • gating mechanism allows for explicit memory deletes and updates.
  • But... the hidden state of RNN is very hard to understand + the long term memory training is still very difficult.

• Instead of using a recurrent matrix to retain information through time, why not build a memory directly?

• The model is then trained to learn how to operate effectively with the memory component. A new kind of learning.
Memory Networks

- Class of models that combine large memory with learning component that can read and write to it.

- Most ML has limited memory which is more-or-less all that’s needed for “low level” tasks e.g. object detection.

- **Motivation**: long-term memory is required to read a story (or watch a movie) and then e.g. answer questions about it.

- **Dataset**: They study this by building a simple simulation to generate “stories”. Also try on some real QA data.
Example of data:

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
James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.

One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.

After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

Q: What did James pull off of the shelves in the grocery store?
A) pudding B) fries C) food D) splinters

Q: Where did James go after he went to the grocery store?
A) his deck B) his freezer C) a fast food restaurant D) his room
James the Turtle was always getting in trouble. Sometimes he’d reach into the freezer and empty out all the food. Other times he’d sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.

Problems: ... it’s hard for this data to lead us to design good ML models ...

1) Not enough data to train on (660 stories total). What ends up happening is that shallow methods perform similarly to fancy ones.

2) If we get something wrong we don’t really understand why: every question potentially involves a different kind of reasoning, our model has to do a lot of different things.

Solution: focus on simpler (toy) subtasks where we can generate data to check what the models we design can and cannot do.

Q: What did James pull off of the shelves in the grocery store?
A) pudding  B) fries  C) food  D) splinters

Q: Where did James go after he went to the grocery store?
A) his deck  B) his freezer  C) a fast food restaurant  D) his room
Simulation: Basic Commands

- go <place>
- get <object>
- get <object1> from <object2>
- put <object1> in/on <object2>
- give <object> to <person>
- drop <object>
- look
- inventory
- examine <object>

Commands only for "gods" (superusers):

- create <object>
- set <object1> <relation> <object2>  - change the graph that represents the world
Example QA in bAbI world.

**Dataset in simulation command format.**

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>antoine go kitchen</td>
</tr>
<tr>
<td>antoine get milk</td>
</tr>
<tr>
<td>antoine go office</td>
</tr>
<tr>
<td>antoine drop milk</td>
</tr>
<tr>
<td>antoine go bathroom</td>
</tr>
<tr>
<td>where is milk? (A: office)</td>
</tr>
<tr>
<td>where is antoine? (A: bathroom)</td>
</tr>
</tbody>
</table>

**Dataset after adding a simple grammar.**

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antoine went to the kitchen.</td>
</tr>
<tr>
<td>Antoine picked up the milk.</td>
</tr>
<tr>
<td>Antoine travelled to the office.</td>
</tr>
<tr>
<td>Antoine left the milk there.</td>
</tr>
<tr>
<td>Antoine went to the bathroom.</td>
</tr>
<tr>
<td>Where is the milk now? (A: office)</td>
</tr>
<tr>
<td>Where is Antoine? (A: bathroom)</td>
</tr>
</tbody>
</table>
Task (1) Factoid QA with Single Supporting Fact ("where is actor")

The first task consists of questions where a single supporting fact, previously given, provides the answer.

They test the simplest case of this, by asking for the location of a person.

A small sample of the task is thus:

John is in the playground.
Bob is in the office.
Where is John? A: playground

It can be considered the simplest case of some real world QA datasets such as in Fader et al., ‘13.

Control the difficulty by the distance in the past of the supporting fact, and how many other irrelevant facts there are.
Task (1) Factoid QA with Single Supporting Fact ("where is actor")

The first task consists of questions where a single supporting fact, previously given, provides the answer.

They test the simplest case of this, by asking for the location of a person.

A small sample of the task is thus:

John is in the playground.  
Bob is in the office.  
Where is John? A: playground

It can be considered the simplest case of some real world QA datasets such as in Fader et al., ’13.

Control the difficulty by the distance in the past of the supporting fact, and how many other irrelevant facts there are.
(2) Factoid QA with Two Supporting Facts ("where is actor+object")

A harder task is to answer questions where two supporting statements have to be chained to answer the question:

<table>
<thead>
<tr>
<th>John is in the playground.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob is in the office.</td>
</tr>
<tr>
<td>John picked up the football.</td>
</tr>
<tr>
<td>Bob went to the kitchen.</td>
</tr>
<tr>
<td>Where is the football?</td>
</tr>
<tr>
<td>Where was Bob before the kitchen?</td>
</tr>
</tbody>
</table>
(2) Factoid QA with Two Supporting Facts

(“where is actor+object”)

A harder task is to answer questions where two supporting statements have to be chained to answer the question:

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: playground  
Where was Bob before the kitchen?  A: office

To answer the first question Where is the football? both John picked up the football and John is in the playground are supporting facts.
Large QA: Reverb Dataset in (Fader et al., 13)

- 14M statements, stored as (subject, relation, object) triples. Triples are REVERB extractions mined from ClueWeb09.

- Statements cover diverse topics:
  - (milne, authored, winnie-the-pooh)
  - (sheep, be-afraid-of, wolf), etc...

- Weakly labeled QA pairs and 35M paraphrased questions from WikiAnswers:
  - "Who wrote the Winnie the Pooh books?"
  - "Who is poohs creator?"

* also worked on the WebQuestions dataset
Memory Networks

MemNNs have four component networks (which may or may not have shared parameters):

**I**: (input feature map) this converts incoming data to the internal feature representation.

**G**: (generalization) this updates memories given new input.

**O**: this produces new output (in feature representation space) given the memories.

**R**: (response) converts the output O into the response format desired. For example, a textual response or an action.
First MemNN Implementation

- **I** (input): no conversion, keep original text $x$.
- **G** (generalization): stores $I(x)$ in next available slot $m_N$.
- **O** (output): Loops over all memories $k=1$ or 2 times:
  - **1\textsuperscript{st} loop max**: finds best match $m_{\text{max}}$ with $x$.
  - **2\textsuperscript{nd} loop max**: finds best match $m_{\text{max2}}$ with $(x, m_{\text{max}})$.
  - The output $o$ is represented with $(x, m_{\text{max}}, m_{\text{max2}})$.
- **R** (response): ranks all words in the dictionary given $o$ and returns best single word. *(OR: use a full RNN here)*
Some Extensions

Some options and extensions:

- **If the input is at the character or word level** one could group inputs (i.e. learn to segment the input into chunks) and put each chunk in a memory slot.

- **Use an RNN for module R** to give true responses.

- **If the memory is huge** (e.g. Wikipedia) we need to organize the memories. Solution: **S** can hash the memories to store in buckets (topics). Then, **G** and **O** don’t operate on all memories.

- **If the memory is full**, there could be a way of removing one it thinks is most useless; i.e. it “forgets” somehow. That would require a scoring function of the utility of each memory.

Explorations of the first 3 points can be found in the paper.
Matching function

- For a given $Q$, we want a good match to the relevant memory slot(s) containing the answer, e.g.:

$\text{Match}(\text{Where is the football ?}, \text{John picked up the football})$

- They use a $q^T U^T U d$ embedding model with word embedding features:
  - $LHS \text{ features: } Q: \text{Where Q:is Q:the Q:football Q:?}$
  - $RHS \text{ features: } D: \text{John D:picked D:up D:the D:football QDMatch:the QDMatch:football}$

$(QDMatch:football \text{ is a feature to say there’s a Q&A word match, which can help.})$

The parameters $U$ are trained with a margin ranking loss: supporting facts should score higher than non-supporting facts.
Matching function: 2nd hop

- On the 2\textsuperscript{nd} hop we match question & 1\textsuperscript{st} hop to new fact:

  \textbf{Match}( [ Where is the football ?, John picked up the football ],
  
  John is in the playground )

- We use the same \textsuperscript{T}U\textsuperscript{T}U\textsuperscript{d} embedding model:
  - \textit{LHS features}: Q:Where Q:is Q:the Q:football Q:? Q2: John Q2:picked Q2:up Q2:the Q2:football
  - \textit{RHS features}: D:John D:is D:in D:the D:playground QDMatch:the QDMatch:is .. Q2DMatch:John

- We also need time information for bAbI simulation.

They tried adding absolute time differences (between two memories) as a feature: tricky to get to work.
Objective Function

given questions, answers, as well as supporting sentences. minimize over parameters $U_O, U_R$

for the 1st supporting fact

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

for the 2nd supporting fact

$$\sum_{\bar{f} \neq f_2} \max(0, \gamma - s_O([x, m_{o1}], f_2)) + s_O([x, m_{o1}], \bar{f}')) +$$

for the finding the response

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_{o1}, m_{o2}], r) + s_R([x, m_{o1}, m_{o2}], \bar{r}'))$$

$S_O$ - Matching function for the Output component
$S_R$ - Matching function for the Response component
x – input question
$m_{o1}$ – first true supporting memory (fact)
$m_{o2}$ – first true second supporting memory (fact)
r – the response

a very common max-margin approach: True facts and response $m_{o1}, m_{o2}$ and r should have higher scores that other facts and response by a given margin
Comparing triples

- Seems to work better if we compare triples:
  - Match(Q,D,D’) returns < 0 if D is better than D’
  - returns > 0 if D’ is better than D

We can loop through memories, keep best $m_i$ at each step.

Now the features include relative time features:

$L.H.S$: same as before

$R.H.S$: features(D)  D_before_Q: 0 or 1

features(D’)  D’_before_Q: 0 or 1  D_before_D’: 0 or 1
• 14M statements stored in the memNN memory.
• k=1 loops MemNN, 128-dim embedding.
• R response simply outputs top scoring statement.
• Time features are not necessary, hence not used.
• Also tried adding bag of words (BoW) features.

<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., 2014)</td>
<td>0.73</td>
</tr>
<tr>
<td>MemNN</td>
<td>0.72</td>
</tr>
<tr>
<td>MemNN (with BoW features)</td>
<td>0.82</td>
</tr>
</tbody>
</table>
Fast QA on Reverb data

- Scoring all 14M candidates in the memory is slow.
- Use speedups via hashing of $S$ and $O$ as mentioned earlier:
  - Hashing via words (essentially: inverted index)
  - Hashing via k-means in embedding space ($k=1000$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Embedding</th>
<th>Embed+BoW</th>
<th>candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemNN (no hashing)</td>
<td>0.72</td>
<td>0.82</td>
<td>14M</td>
</tr>
<tr>
<td>MemNN (word hash)</td>
<td>0.63</td>
<td>0.68</td>
<td>13k (1000x)</td>
</tr>
<tr>
<td>MemNN (clust hash)</td>
<td>0.71</td>
<td>0.80</td>
<td>177k (80x)</td>
</tr>
</tbody>
</table>
bAbI (Simulation) Experiment 1

- 10k sentences. (Actor: only ask questions about actors.)
- Difficulty: how many sentences in the past when entity mentioned.
- Fully supervised (supporting sentences are labeled).
- Compare RNN (no supervision)
- and MemNN hops $k = 1$ or $2$, & with/without time features.

<table>
<thead>
<tr>
<th>Method</th>
<th>Difficulty 1</th>
<th>Difficulty 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>actor</td>
<td>actor+object</td>
</tr>
<tr>
<td>RNN</td>
<td>0%</td>
<td>42%</td>
</tr>
<tr>
<td>MemNN $k = 1$</td>
<td>10%</td>
<td>81%</td>
</tr>
<tr>
<td>MemNN $k = 1$ (+time)</td>
<td>0%</td>
<td>27%</td>
</tr>
<tr>
<td>MemNN $k = 2$ (+time)</td>
<td>0%</td>
<td>0.05%</td>
</tr>
</tbody>
</table>

*Difficulty 5 -- Max mem. sz. required: 65   Average mem. sz. required: 9*
Example test story + predictions:

Antoine went to the kitchen. Antoine got the milk. Antoine travelled to the office. Antoine dropped the milk. Sumit picked up the football. Antoine went to the bathroom. Sumit moved to the kitchen.

- where is the milk now? A: office
- where is the football? A: kitchen
- where is Antoine? A: bathroom
- where is Sumit? A: kitchen
- where was Antoine before the bathroom? A: office
Unsegmented setup; R module is an RNN

- 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
Dealing with new words

Sometimes words come along you’ve never seen before, e.g. a new name:

`Here,' said Elrond, turning to Gandalf, `is Boromir, a man from the South. He arrived in the grey morning, and seeks for counsel”

Our approach: incorporate missing word prediction in same model.

Boromir is represented as unknown with left and right context words, features shown in bold:

Then Boromir goes to the office

We can learn this with the original training data using a kind of “dropout”: sometimes we pretend we don’t have a word’s embedding, and represent it as above instead.
A MemNN that has never seen Frodo, Bilbo or the ring before..

The “story” told to the model after training:

“Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring. Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring. Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died. Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.”

MemNN’s answers to some questions:

- Where is the ring? A: Mount-Doom
- Where is Bilbo now? A: Grey-havens
- Where is Frodo now? A: Shire
AMemNN multitasked on bAbI data and Reverb QA data

The “story” told to the model after training:

Antoine went to the kitchen. Antoine picked up the milk. Antoine travelled to the office.

MemNN’s answers to some questions:

- Where is the milk? A: office
- Where was Antoine before the office? A: kitchen
- Where does milk come from? A: milk come from cow
- What is a cow a type of? A: cow be female of cattle
- Where are cattle found? A: cattle farm become widespread in brazil
- What does milk taste like? A: milk taste like milk
- What does milk go well with? A: milk go with coffee
What’s next?

- Make it harder/add more stuff, e.g. “he went..”, “Frodo and Sam”, etc.!!!

- MemNNs that reason with more than 2 supporting memories.

- Weakly supervised?

- Ask questions? Say statements? Perform actions?

- MCTest reading comprehension data (Richardson et al.)

- Do MemNN ideas extend to other ML tasks and model variants...? [A: probably, yes!].