Memory Networks

By Jason Weston, Sumit Chopra and Antoine Bordes
Presented by Dongming Lei

Outline

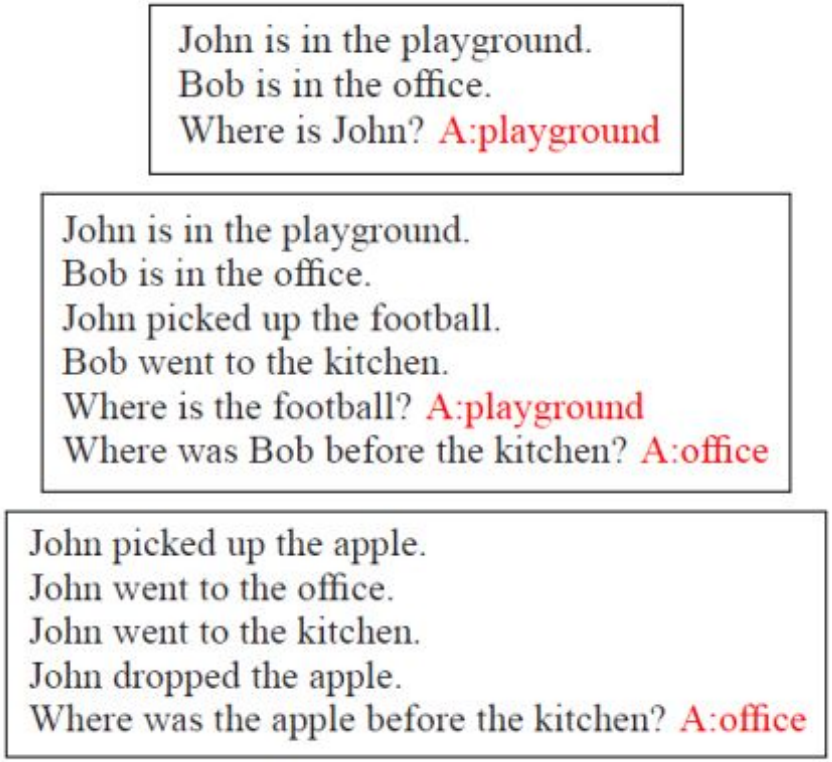
- Introduction and Related Work
- Memory Networks
- Basic QA Model using Memory Networks
- Extensions to the Basic Model
- Experiments
  - Large Scale QA
  - Simulated World QA
- Conclusion and Future Work

Introduction

Toy Example of QA

To answer the question:

- Understanding the question and the story
- Finding the supporting facts for the question
- Generating an answer based on supporting facts



Related Work

- 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:
  - Representing questions and answers as embeddings via neural sentence Models
  - Learning matching models and embeddings by question-answer pairs
- How about memory & reasoning?

Memory Networks

- Memory Network: class of models that combine long-term memory with learning component that can read and write to it.
- **Motivation**: long-term memory is required to read a story (or watch a movie) and then e.g. answer questions about it.
- This study is done using experiments on
  - building a simple simulation to generate ``stories''
  - real large-scale QA data

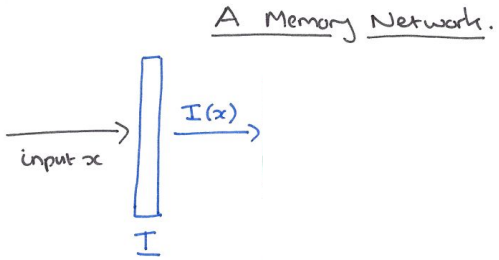
Memory Networks

A **Memory Network** consists of a memory **m**: an array of objects indexed by m_i

and **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**: (output feature map) this produces new output (in feature representation space) given the memories.
- **R**: (response) converts output O into a response seen by the outside world.

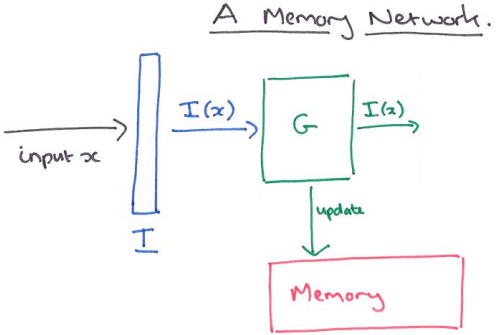
Memory Networks



Given an input x (e.g. a sentence - the statement of a fact or a question, the flow of the model

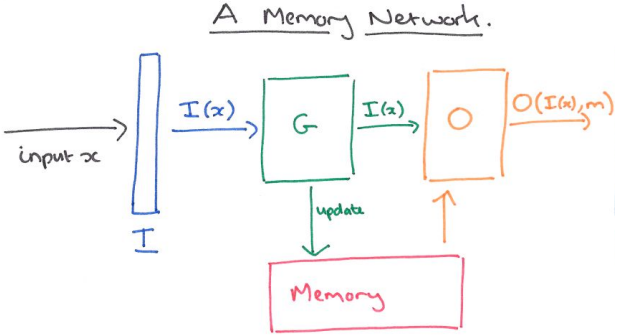
- **I**: (input feature map) Convert x to an internal feature representation I(x)

Memory Networks



- **G**: (generalization) Update memory m_i given the new input: $m_i = G(m_i, I(x), m)$
- Store text in the next available memory slot in its original form
  - $m_N = x, N = N + 1$

Memory Networks



- **O**: (output feature map) compute output features o given the new input and the memory. o = O(I(x), m)
- For example, finding k supporting memories given x

Take $k = 2$ as an example:

$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

The final output o: $[x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}]$

Memory Networks

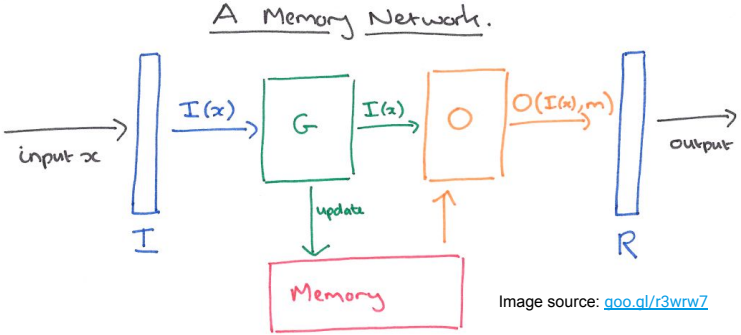


Image source: goo.gl/r3wrw7

- **R:** (response) Finally, decode output features o to give the final response: r = R(o)
- Producing a textual response r

$$r = \arg\max_{w \in W} s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

where W is the word vocabulary

Model - Example

- In order to answer the question x = “Where is the milk now?”
- The O module first scores all memories - all previous seen sentences against x to retrieve the most relevant fact m_{o1}=”Joe left the milk”
- Then, it would search memory again to find the second relevant fact given [x, m_{o1}], that is m_{o2}=”Joe travelled to the office”
- Finally, the R module would score words give [x,m_{o1},m_{o2}] to output r = “office”

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

Model Training

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:

$$\sum_{\bar{f} \neq f_1} \max(0, \gamma - s_O(x, f_1) + s_O(x, \bar{f})) +$$
$$\sum_{\bar{f}' \neq f_2} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], f_2]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}'])) +$$
$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r) + s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r}]))$$

Extension

- Efficient Memory Via Hashing
  - The set of stored memories is very large; Scoring all the memories to find the best supporting one is prohibitively **expensive**
  - Hashing words: a memory mi will only be considered if it shares at least one word with the input I(x)
  - Clustering word embeddings: run K-means to cluster word vectors to K buckets
- Exact Matches
  - Embedding models cannot efficiently use exact word matches due to the low dimensionality
  - Instead, score a pair x, y with $\Phi_x(x)^\top U^\top U \Phi_y(y) + \lambda \Phi_x(x)^\top \Phi_y(y)$
  - “Bag of words” matching score to the learned embedding score with a mixing parameter lambda

Extension

- Modeling Write Time
  - Answering questions about a story: relative order of events is important
  - Option 1: add extra features to encode absolution write time
  - Option 2: 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 Φ embeddings by 3
    - Φt (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_{Ot} (x, y, y') > 0, the model prefers y; otherwise y'

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

Experiments: Large-Scale QA

- Dataset:
  - 14M statements; stored as triples (subject, relation, object)
  - REVERB extractions mined from ClueWeb09 corpus and covers diverse topics
  - Questions: generated from seed patterns, e.g. What is sheep afraid of?
- Results
  - k=1 supporting memory

Method	F1
(Fader et al., 2013)	0.54
(Bordes et al., 2014b)	0.73
MemNN (embedding only)	0.72
MemNN (with BoW features)	0.82

Method	Embedding F1	Embedding + BoW F1	Candidates (speedup)
MemNN (no hashing)	0.72	0.82	14M (0x)
MemNN (word hash)	0.63	0.68	13k (1000x)
MemNN (cluster hash)	0.71	0.80	177k (80x)

Experiments: Simulated World QA

- Dataset
  - Simulation of 4 characters, 3 objects and 5 rooms with characters moving around, picking up and dropping objects.
  - Actions are transcribed into text
  - Difficulty is that multiple statements have to be used to do inference
    - Control the difficulty of the task by setting a limit on the number of time steps in the past the entity they ask the question about was last mentioned
  - 7k statements and 3k questions

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

Experiments: Simulated World QA

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

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

Conclusions and Future Work

- Introduced a class of models, memory networks and showed one instantionation for QA
- Can explore
  - harder QA and open-domain machine comprehension tasks
  - more complex simulation data, such as coreference, involving more verbs and nouns, sentences with more structure and requiring more temporal and causal understanding
  - more sophisticated architectures and sentence representation
  - Weakly supervised setting
  - Other tasks and domains such as vision

Thanks!