Outline

Outline

Authors

- Memory Networks
- **Jason Weston**, Sumit Chopra & Antoine Bordes
- Facebook AI Research
- arXiv.org, 15 Oct 2014 (9 Apr 2015, to ICLR 2015)

- Weakly Supervised Memory Networks
- Sainbayar Sukhbaatar, Arthur Szlam, **Jason Weston**, Rob Fergus
- New York University, Facebook AI Research
- arXiv.org, 31 Mar 2015 (3 Apr 2015)

- Weston, J., Bordes, A., Chopra, S., and Mikolov, T. Towards AI-complete question answering: A set of prerequisite toy tasks. arXiv preprint: 1502.05698, 2015
- Bordes, A., Chopra, S., and Weston, J. Question answering with subgraph embeddings. In Proc. EMNLP, 2014.
- Bordes, A., Weston, J., and Usunier, N. Open question answering with weakly supervised embedding models. ECML-PKDD, 2014.

Introduction

Recall some toy tasks of Question Answering [1]:

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

[1] Weston, J., Bordes, A., Chopra, S., and Mikolov, T. Towards AI-complete question answering: A set of prerequisite toy tasks. arXiv preprint: 1502.05698, 2015 (`http://fb.ai/babi`)

Introduction (cont.)

Simulated World QA:

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

$\rightarrow$ A story, a related question and an answer

Introduction (cont.)

Simulated World QA:

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

$\rightarrow$ 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

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?

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

Memory Networks: General Framework

Components: $(\mathbf{m}, I, G, O, R)$

- **A memory $\mathbf{m}$: an array of objects indexed by $\mathbf{m}_i$**
- **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: $(\mathbf{m}, I, G, O, R)$

- **A memory $\mathbf{m}$: an array of objects indexed by $\mathbf{m}_i$**
- **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 $\mathbf{m}_i$ given the new input: $\mathbf{m}_i = G(\mathbf{m}_i, I(x), \mathbf{m}), \forall i$.
3. Compute output features o given the new input and the memory: $o = O(I(x), \mathbf{m})$.
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 $\mathbf{m}_{H(x)} = I(x)$
  - More sophisticated form: go back and update earlier stored memories based on the new evidence from the current input[4]
  - 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 preforming 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)

[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 $\mathbf{m}_{H(x)} = I(x)$
  - More sophisticated form: go back and update earlier stored memories based on the new evidence from the current input[4]
  - 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 preforming 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] similar to LSTM

MemNN Models for Text

Basic MemNN Model for Text:

- $(\mathbf{m}, I, G, O, R)$

Variants of Basic MemNN Model for Text

- Word Sequences as Input
- Efficient Memory via Hashing
- Modeling Writing Time
- Modeling Preivous 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:
$$\mathbf{m}_N = x, N = N + 1$$
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:
$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$
$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$
The final output o: $[x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}]$
- R: 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

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 repsonse: 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 reponse (words or phrases)

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

Φ_x, Φ_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 $\mathbf{m}$)

MemNNs for Text: Basic Model (cont.)

Scoring Function for the output and repsonse: 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 reponse (words or phrases)

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

Φ_x, Φ_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 $\mathbf{m}$)

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:

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

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

MemNN: Word Sequences as Input

Situation:

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

MemNN: Word Sequences as Input

Situation:

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

$\rightarrow$ Add a segmentation function: sequences $\rightarrow$ sentences

$$seg(c) = W_{seg}^T U_S \Phi_{seg}(c)$$

where c is the input word sequence (BoW using a separate dictionary)

If $seg(c) > \gamma$ (i.e. the margin), this sequence is recognised as a segment.

$\rightarrow$ 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

$\rightarrow$ Exploring hashing tricks to speed up lookup:
hash the input $I(x)$ into one or more buckets and then only score memories $\mathbf{m}_i$ 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 $\mathbf{m}_i$ 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
$\rightarrow$ Take in to account **when** a memory slot was written to

- Add extra features to Φ_x and Φ_y to encode absolute write time
- Learning a function on triples to get relative time order

$$s_{O_t}(x, y, y') = \Phi_x(x)^\top {U_{O_t}}^\top U_{O_t}\Big(\Phi_y(y) - \Phi_y(y') + \Phi_t(x, y, y')\Big)$$

  - extending the dimensionality of all the Φ 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 perfers y; otherwise y'

    $\rightarrow$ 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):
    $\rightarrow$ **stored as memories in the MemNN model**
  - Triples: $REVERB$ 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

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[5]

Dataset:

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

$\rightarrow$ 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.)

	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%

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.

Fred went to the kitchen. Fred picked up the milk. Fred travelled to the office.
Where is the milk ? A: office
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
Where was Fred before the office ? A: kitchen

Outline

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

$\rightarrow$ 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 bAbI[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 bAbI tasks that probe different forms of reasoning and deduction.
- Formal Task Description:
  - For one of the 20 bAbI tasks, we are given P example problems, each having a set of I sentences x_i^p where $I320$; a question sentence q^p and answer a^p.
  - The examples are randomly split into disjoint train and test sets
  - Let the jth word of sentence i be x_{ij} , represented by a one-hot vector of length V (where $|V| = 177$ since the bAbI 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 $m_i \in \mathbb{R}^d$: $m_i = \sum_j Ax_{ij}$
- For the question:
  The question vector q is also embedded via matrix B: $u = \sum_j Bq_j$
- For the match between the question u and each memory m_i:
  The probability vector: $p_i = softmax(u^T m_i) = softmax(q^T B^T \sum_j Ax_{ij})$

OUTPUT Side:

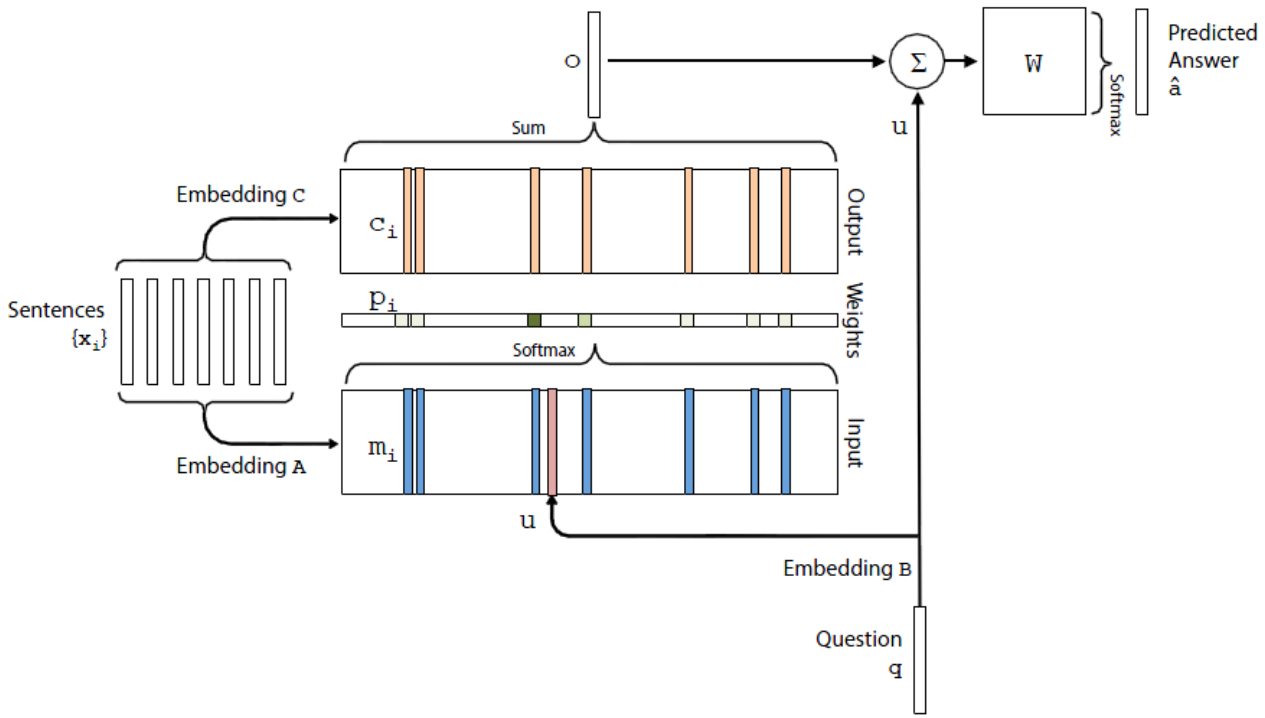
- Each memory vector on the input has a corresponding output vector c_i: $c_i = \sum_j Cx_{ij}$
- The output vector o from the memory: $o = \sum_i p_i c_i = \sum_i \sum_j p_i Cx_{ij}$

MemNN-WSH: Single Layer(cont.)

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 $\hat{a}$: $\hat{a} = softmax(W(o + u))$

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

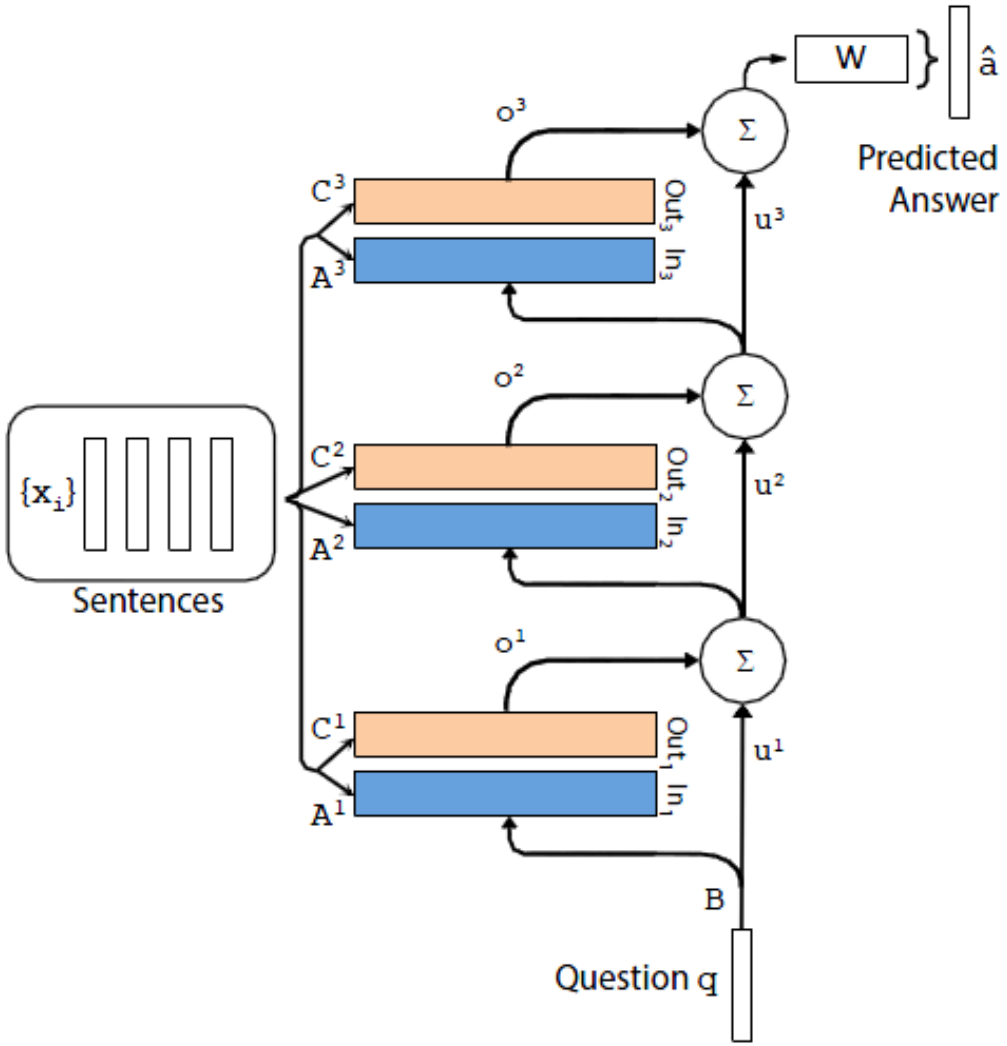


MemNN-WSH: Multiple Layers

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: $\hat{a} = softmax(W(o^K + u^K))$

MemNN-WSH: Multiple Layers(cont.)



MemNN-WSH

Sentence Representation:

- BoW representation: the sum of words $m_i = \sum_j Ax_{ij}$
- PE representation: Encoding the position of words within the sentence (used for questions, memory inputs and memory outputs)

$$m_i = \sum_j l_j \cdot Ax_{ij}$$

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: $m_i = \sum_j Ax_{ij} + T_A(i)$
- Augment the output in the same way: $c_i = \sum_j Cx_{ij} + T_C(i)$
- 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)

	Supervised		Weakly Supervised								
	Baseline	Ours	Baseline		Ours						
Task	MemNN	PE	LSTM	MemNN WSH	BoW	PE	PE LS	PE LS RN	PE LS RN RNN	PE LS joint	PE LS RNN joint
1: 1 supporting fact	0.0	0.0	50.0	0.1	0.3	0.0	0.1	0.0	0.0	0.0	0.0
2: 2 supporting facts	0.0	11.0	80.0	42.8	14.3	12.5	11.5	8.0	58.0	13.9	7.2
3: 3 supporting facts	0.0	14.7	80.0	76.4	60.3	57.7	43.8	36.8	70.0	23.0	21.4
4: 2 argument relations	0.0	21.8	39.0	40.3	32.8	13.7	16.1	3.9	0.3	3.9	11.8
5: 3 argument relations	2.0	5.4	30.0	16.3	16.9	11.4	12.5	10.3	2.7	11.9	1.2
6: yes/no questions	0.0	12.6	52.0	51.0	7.3	7.4	7.4	7.0	8.9	1.8	1.0
7: counting	15.0	20.2	51.0	36.1	17.9	16.2	18.0	15.8	15.2	13.3	10.2
8: lists/sets	9.0	18.5	55.0	37.8	10.6	10.4	10.5	8.0	9.1	9.3	4.5
9: simple negation	0.0	16.2	36.0	35.9	19.6	15.7	10.1	11.1	16.9	2.0	0.2
10: indefinite knowledge	2.0	34.8	56.0	68.7	18.6	15.2	16.8	10.6	16.8	4.8	0.2
11: basic coherence	0.0	0.3	38.0	30.0	4.5	2.9	0.2	0.5	10.6	0.7	2.7
12: conjunction	0.0	0.1	26.0	10.1	0.2	0.0	0.0	0.0	0.0	0.1	0.0
13: compound coherence	0.0	0.1	6.0	19.7	8.9	8.2	0.4	0.1	6.6	0.2	0.5
14: time reasoning	1.0	0.2	73.0	18.3	2.6	2.3	1.6	0.8	15.5	3.7	3.3
15: basic deduction	0.0	0.0	79.0	64.8	19.4	0.0	0.0	0.0	0.0	0.0	0.0
16: basic induction	0.0	0.3	77.0	50.5	49.8	52.2	1.4	1.3	50.5	2.0	48.6
17: positional reasoning	35.0	54.9	49.0	50.9	50.9	48.0	48.6	50.8	48.3	44.3	41.2
18: size reasoning	5.0	38.0	48.0	51.3	48.8	10.3	11.9	8.9	9.7	9.0	8.9
19: path finding	64.0	83.2	92.0	100.0	89.5	87.4	86.2	84.0	87.7	88.4	89.5
20: agent's motivation	0.0	0.0	9.0	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mean error (%)	6.7	16.6	51.3	40.2	23.7	18.6	14.9	12.9	21.3	11.6	12.6
Failed tasks (err. $>$ 5%)	4	12	20	18	15	14	12	11	14	8	8

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.)

Task	Supervised Baseline	Weakly Supervised						
		Baseline	Ours					
	MemNN	MemNN WSH	BoW	PE	PE LS	PE LS RNN	PE LS joint	PE LS RNN joint
1: 1 supporting fact	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
2: 2 supporting facts	0.0	39.6	0.5	0.4	0.3	0.3	0.5	0.6
3: 3 supporting facts	0.0	79.5	14.2	13.5	12.1	6.9	8.6	14.3
4: 2 argument relations	0.0	36.6	32.1	0.0	0.0	0.0	0.0	0.0
5: 3 argument relations	0.3	21.1	11.9	0.4	0.4	0.2	7.6	0.5
6: yes/no questions	0.0	49.9	0.2	0.2	0.4	0.0	0.0	0.0
7: counting	3.3	35.1	8.3	3.3	4.2	2.1	5.3	3.8
8: lists/sets	1.0	42.7	1.0	1.7	1.5	0.6	2.1	0.8
9: simple negation	0.0	36.4	0.3	0.6	1.6	0.5	0.0	0.0
10: indefinite knowledge	0.0	76.0	1.4	0.9	2.0	1.0	0.4	0.0
11: basic coherence	0.0	25.3	0.0	0.0	0.0	0.0	0.0	0.3
12: conjunction	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13: compound coherence	0.0	12.3	0.0	0.0	0.0	0.0	0.0	0.0
14: time reasoning	0.0	8.7	0.1	0.0	0.0	0.0	0.0	0.0
15: basic deduction	0.0	68.8	12.3	0.0	0.0	0.0	0.0	0.0
16: basic induction	0.0	50.9	47.6	48.0	0.0	47.5	0.0	46.9
17: positional reasoning	24.6	51.1	46.6	39.8	36.9	35.1	41.2	41.0
18: size reasoning	2.1	45.8	37.4	8.4	6.4	8.0	8.1	8.1
19: path finding	31.9	100.0	75.0	66.1	64.9	25.6	73.2	67.1
20: agent's motivation	0.0	4.1	0.0	0.0	0.0	0.0	0.0	0.0
Mean error (%)	3.2	39.2	14.4	9.2	6.5	6.4	7.3	9.2
Failed tasks (err. $> 5\%$)	2	17	9	5	4	5	6	5

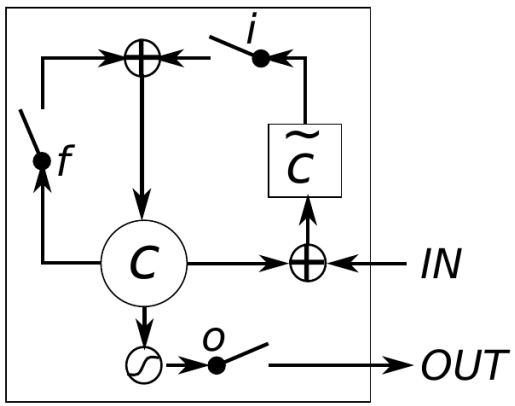
Table 2: Test error rates (%) on the 20 bAbI tasks for models using 10k training examples. Key: BoW = bag-of-words representation; PE = position encoding representation; LS = linear start training; RNN = RNN-style layer-wise weight tying (if not stated, adjacent weight tying is used); joint = joint training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

Outline

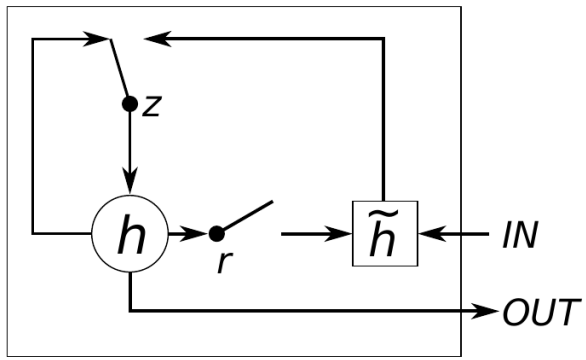
Appendix: Gated Recurrent Neural Networks

LSTM & GRU

Chung, J.,et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv'14.



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

LSTM Unit:

$$h_t^j = o_t^j \tanh(c_t^j)$$
$$o_t^j = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t)^j$$
$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j$$
$$\tilde{c}_t^j = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})^j$$
$$f_t^j = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_t)^j$$
$$i_t^j = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_t)^j$$

GRU Unit:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j$$
$$z_t^j = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})^j$$
$$\tilde{h}_t^j = \tanh(W \mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))^j$$
$$r_t^j = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})^j$$