Learning to Query, Reason, and Answer Questions On Ambiguous Texts

Xiaoxiao Guo, Tim Klinger, Clemens Rosenbaum, Joseph P. Bigus, Murray Campbell, Ban Kawas, Kartik Talamadupula, Gerald Tesauro, Satinder Singh

University of Michigan, IBM Research, UMass Amherst

ICLR 2017

Presenter: Jack Lanchantin

Outline

1. Introduction
    - Background
    - This paper: QRAQ
2. QRAQ
3. Model
    - Control Loop
    - baseRL
    - impRL
    - Policy Gradient & Reward Function
4. Data, Training, and Results

Outline

Motivation

- Human conversation is incomplete, ambiguous and full of extraneous detail
- Conversational agents must be able to reason in the presence of missing or unclear info

Previous Datasets

- bAbI & children's book: answer questions about short stories
- Task-oriented dialog systems: answer questions to find restaurants or movies (slot filling)

Outline

QRAQ (Query, Reason, and Answer Questions) Dataset

1. Simulator provides a story and a question to the agent, with some of the entities replaced by variables

QRAQ (Query, Reason, and Answer Questions) Dataset

1. Simulator provides a story and a question to the agent, with some of the entities replaced by variables
2. The agent must be able to decide whether it has enough info to answer the question

QRAQ (Query, Reason, and Answer Questions) Dataset

1. Simulator provides a story and a question to the agent, with some of the entities replaced by variables
2. The agent must be able to decide whether it has enough info to answer the question
3. If the agent cannot answer the question by reasoning alone, it must learn to query the simulator for a variable value

QRAQ Problem

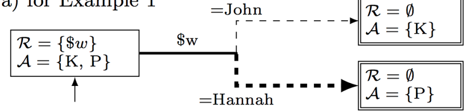
C1. Hannah is in the garden.
C2. $u is Emma.
C3. $u is in the garden.
C4. The gift is in the garden.
C5. John is in the kitchen.
C6. The ball is in the kitchen.
C7. The skateboard is in the kitchen
E1. Hannah picks up the gift.
E2. John picks up $x.
E3. $v goes from the garden to the kitchen.
E4. $w walks from the kitchen to the patio.
E5. Having left the garden, $u goes to the patio.
Q. Where is the gift?
GT. $v = Hannah; $w = Hannah; Answer = Patio

$$C_1, C_2, \ldots : \text{ Context}$$
$$E_1, E_2, \ldots : \text{ Events}$$
$$Q: \text{ Question}$$
$$GT: \text{ Ground Truth}$$

QRAQ Query Graph

C1. Hannah is in the garden.
C2. \$u is Emma.
C3. \$u is in the garden.
C4. The gift is in the garden.
C5. John is in the kitchen.
C6. The ball is in the kitchen.
C7. The skateboard is in the kitchen
E1. Hannah picks up the gift.
E2. John picks up \$x.
E3. \$v goes from the garden to the kitchen.
E4. \$w walks from the kitchen to the patio.
E5. Having left the garden, \$u goes to the patio.
Q. Where is the gift?
GT. \$v = Hannah; \$w = Hannah; Answer = Patio

(a) for Example 1

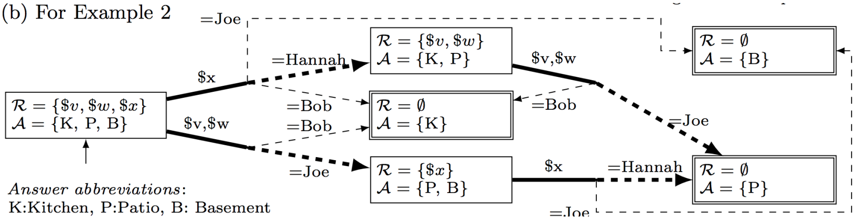


Legend: Nodes represent Agent state = $(\mathcal{R}, \mathcal{A})$.
$\mathcal{R}$ = relevant variables
$\mathcal{A}$ = possible answers to the challenge question.
Each arrow represents a variable query action (solid part) and observed outcomes (dashed part).
Multiple variables on a query-edge means that they have the same outcome set.
Double-bordered nodes have a unique challenge question answer.
A thick line denotes a ground-truth-path.

QRAQ Query Graph

C1. Joe is in the kitchen.
C2. Bob is in the kitchen.
C3. Hannah is in the patio.
E1. \$v goes from the kitchen to the garden.
E2. \$w goes from the garden to the patio.
E3. \$x goes from the patio to the basement.
Q. Where is Joe?
GT. \$v = Joe; \$w = Joe; \$x = Hannah; answer = Patio

(b) For Example 2



Outline

Control Loop

1. **Initialization**: Question vector c and memory matrix S_0 are initialized

Control Loop

1. **Initialization**: Question vector c and memory matrix S_0 are initialized
2. **Action Selection**: Policy $\pi(a|S_t, c)$ maps memory matrix S_t and question c, into a distribution over actions. Action a could be either a query for a variable or a final answer

Control Loop

1. **Initialization**: Question vector c and memory matrix S_0 are initialized
2. **Action Selection**: Policy $\pi(a|S_t, c)$ maps memory matrix S_t and question c, into a distribution over actions. Action a could be either a query for a variable or a final answer
3. **Variable Query and Memory Update**: If action is a query, simulator provides the true value v_t for the variable in action a_t. All occurrences of variable in action a_t in the memory S_t are replaced with the true value v_t : $S_{t+1} = S_t[a_t \rightarrow v_t]$

Control Loop

1. **Initialization**: Question vector c and memory matrix S_0 are initialized
2. **Action Selection**: Policy $\pi(a|S_t, c)$ maps memory matrix S_t and question c, into a distribution over actions. Action a could be either a query for a variable or a final answer
3. **Variable Query and Memory Update**: If action is a query, simulator provides the true value v_t for the variable in action a_t. All occurrences of variable in action a_t in the memory S_t are replaced with the true value v_t : $S_{t+1} = S_t[a_t \rightarrow v_t]$
4. **Final Answer Generation and Termination**: If the action is an answer, task terminates and a reward is generated based on correctness

Outline

baseRL: End-to-End Memory Net Based Policy Learner

- Maps the memory matrix, S, and the challenge question representation, c, into an action distribution.
- S^{ij} is dictionary index of j^{th} word in the i^{th} sentence
- c_i is the dictionary index of the i^{th} word in the question

$$m_i = \sum_j l_j \circ A[S^{ij}] \tag{1}$$

$$q = \sum_j l_j \circ A[c_j] \tag{2}$$

$A \in \mathbb{R}^{d \times N}$, $A[k]$ returns the k-th column vector (sentence), d is embedding dimension, N is the dictionary size,
$l_j^k = (1 - j/J)(k/d)(1 - 2j/J)$, with J being the number of words in the sentences

baseRL: End-to-End Memory Net Based Policy Learner

$$m_i = \sum_j l_j \circ A[S^{ij}] \tag{3}$$

$$q = \sum_j l_j \circ A[c_j] \tag{4}$$

Output vector from reading $\{m_i\}$ after the k^{th} hop is u_k ($u_0 = q$):

$$\begin{align} u_k &= tanh(H(o_k + u_{k-1})) \tag{5} \\ o_k &= \sum_i p_i^k m_i \tag{6} \\ p_i^k &= softmax(u_{k-1}^T m_i) \tag{7} \\ & \tag{8} \end{align}$$

baseRL: End-to-End Memory Net Based Policy Learner

Query Network Output. Since the problems have at most one variable per sentence, the distribution can be converted into the distribution over sentences:

$$\pi_Q^i = softmax(u_K^T m_i) \tag{9}$$

Answer Network Output. The final output of the policy module is a distribution over potential answers:

$$\pi_A = softmax(Wu_K + b) \tag{10}$$

baseRL $\rightarrow$ impRL

- baseRL: Final action-distribution output only conditions on the last hop output
- impRL: computes the final action-distribution-output over all memory hop outputs

Outline

impRL: Improved End-to-End Memory Net Based Policy Learner

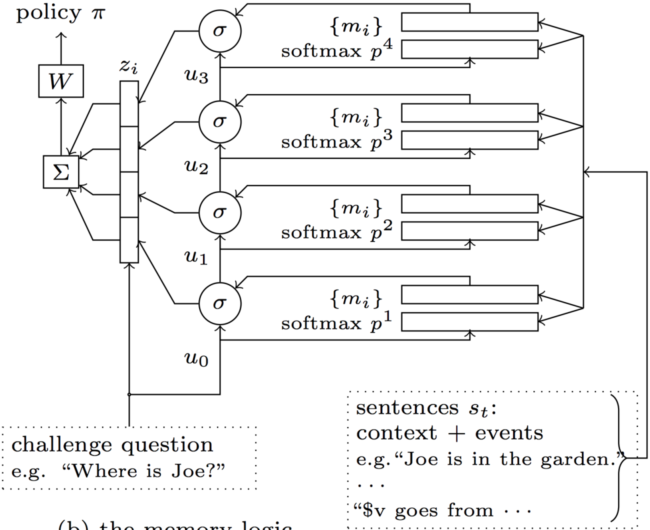
$$\pi_Q^i = softmax(u^T m_i) \tag{11}$$

$$\pi_A = softmax(Wu + b) \tag{12}$$

$$u = \sum_j z_j u_j \tag{13}$$

$$z_j = softmax(q^T u_j) \tag{14}$$

impRL: Improved End-to-End Memory Net Based Policy Learner



(b) the memory logic

Outline

Policy Gradient & Reward Function

- +Reward when the action is the correct answer
- -Reward when the action is a wrong answer
- -Reward when the action is a query to a variable
- Objective function is to optimize the expected cumulative reward over M training problem instances

$$\sum_{m=1}^{M} \mathbb{E}\{\sum_{t} r_t^m\} \tag{15}$$

- GPOMDP (Weaver & Tao (2001)) is used to calculate the policy gradient

Data

- 4 Types
- 107,000 QRAQ problems in each type
  - 100,000 training
  - 2,000 testing
  - 5,000 validation

Data

- **(Loc)**: Context and events describe locations and movements of people in rooms. Questions are about the location of a specific person
- **(+obj)**: Adds objects to (Loc)
- **(+alias)**: Adds aliases to (Loc) Some of them are defined in the context.
- **(+par)**: Substitutes sentences with semantically equivalent paraphrases in (Loc)

Curriculum Learning

- First encourage the agent to query variables by assigning positive rewards for querying any variable
- After convergence under this initial reward function, switch to the true reward function that assigns a negative reward for querying a variable to reduce the number of unnecessary queries
- +1 for correct final answers, -5 for wrong final answers, query reward +/-0.005

Supervised Learning (SL) Baseline

- Upper-bound on achievable reinforcement learning performance
- Relevant variables and (when appropriate) the correct final answers are provided at each turn, and the cross entropy is used to optimize

Accuracy Metrics

Trajectory: a sequence of variable queries followed by an answer

1. **Answer-accuracy**: proportion of trajectories in which the agent answered correctly
2. **Trajectory-accuracy**: proportion of trajectories in which the agent queried only relevant variables then answered correctly
3. **Trajectory-completeness**: proportion of trajectories in which the agent queried all and only relevant variables before answering correctly
4. **Query accuracy**: the proportion of correct queries among all queries made in any trajectory

Results

Table 1: Datasets. The first 7 rows give statistics on the datasets themselves. The last 8 rows show results for answer accuracy (AnsAcc), trajectory accuracy (TrajAcc), trajectory completeness (TrajCmpl) and query accuracy (QryAcc) for the impRL and baseRL agents on the respective datasets. The middle 8 rows show results for the supervised learning agents.

Data Set	*(Loc)*					*(+obj)*	*(+alias)*	*(+par)*
#names in vocab	5	20	10	20	20	20	20	20
#var in vocab	5	20	10	20	20	20	20	20
#sentence/prob.	5-6	5-6	7-10	15-20	19-23	7-10	10-12	10-12
#var/prob.	0-2	0-2	0-2	0-3	5-10	5-10	0-5	0
depth	0-2	0-2	0-2	0-2	4-9	0-2	0-5	0
avg. depth	0.817	0.872	0.558	0.459	5.087	0.543	1.066	-
sum(depth) / sum(#var)	0.734	0.748	0.313	0.204	0.703	0.404	0.310	-
AnsAcc in %; impSL	99.9	99.5	92.1	95.3	91.4	95.9	90.7	99.8
AnsAcc in %; baseSL	99.9	99.2	92.3	92.4	90.2	95.5	86.6	98.8
TrajAcc in %; impSL	99.6	98.9	90.2	88.4	85.3	95.2	86.7	-
TrajAcc in %; baseSL	98.9	98.7	90.3	86.5	83.3	94.9	85.3	-
TrajCmpl in %; impSL	99.5	98.8	89.9	85.6	80.9	94.9	83.6	-
TrajCmpl in %; baseSL	98.7	98.7	90.0	83.5	78.7	94.6	82.9	-
QryAcc in %; impSL	99.5	99.2	96.4	84.6	93.5	97.7	93.7	-
QryAcc in %; baseSL	98.7	99.3	96.3	85.5	92.7	97.5	97.0	-
AnsAcc in %; impRL	99.1	94.4	86.5	89.0	64.2	81.1	75.7	96.9
AnsAcc in %; baseRL	98.4	95.0	88.4	88.2	54.6	79.6	69.7	97.2
TrajAcc in %; impRL	94.5	90.9	61.9	52.0	45.1	74.9	63.2	-
TrajAcc in %; baseRL	94.8	90.4	63.6	52.5	35.7	73.9	60.5	-
TrajCmpl in %; impRL	94.5	88.7	55.8	46.9	37.8	61.8	56.4	-
TrajCmpl in %; baseRL	94.6	89.5	59.9	47.4	28.3	61.2	54.5	-
QryAcc in %; impRL	94.3	95.4	49.2	32.1	80.0	69.6	77.0	-
QryAcc in %; baseRL	95.5	94.1	54.6	32.0	76.5	71.0	79.6	-

Conclusion

- New dataset, QRAQ, for reasoning under insufficient information
- First to formulate these types of QA problems in the RL format