Neural Networks 106 (2018) 42–49

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Learning to activate logic rules for textual reasoning

Yiqun Yao [a,b,c], Jiaming Xu [a,b,*], Jing Shi [a,b,c], Bo Xu [a,b,c,d]

[a] Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China
[b] Research Center for Brain-inspired Intelligence, CASIA, China
[c] University of Chinese Academy of Sciences, China
[d] Center for Excellence in Brain Science and Intelligence Technology, CAS, China

ARTICLE INFO

Article history:
Received 20 March 2018
Received in revised form 11 June 2018
Accepted 22 June 2018
Available online 3 July 2018

Keywords:
Natural language reasoning
Memory networks
Image schema
Logic rules
Reinforcement learning

ABSTRACT

Most current textual reasoning models cannot learn human-like reasoning process, and thus lack interpretability and logical accuracy. To help address this issue, we propose a novel reasoning model which learns to activate logic rules explicitly via deep reinforcement learning. It takes the form of Memory Networks but features a special memory that stores relational tuples, mimicking the "Image Schema" in human cognitive activities. We redefine textual reasoning as a sequential decision-making process modifying or retrieving from the memory, where logic rules serve as state-transition functions. Activating logic rules for reasoning involves two problems: *variable binding* and *relation activating*, and this is a first step to solve them jointly. Our model achieves an average error rate of 0.7% on bAbI-20, a widely-used synthetic reasoning benchmark, using less than 1k training samples and no supporting facts.

1. Introduction

One of the long term goals of artificial intelligence is precise reasoning. Reasoning performance is essential in natural language processing (NLP) tasks, i.e. machine translation, field question answering and text games. Although current models can learn a lot of language phenomena from large data, some subtle corner cases are often ignored: counting, copying, positional reasoning, antonyms, etc. Most of these problems need some external knowledge of logics to be well solved, especially in specific domains and small tasks. Even with large data, it is hard to learn human logic perfectly because models tend to learn biases from dataset instead of truly understand the reasoning process behind them.

The bAbI-20 dataset (Weston, Bordes et al., 2015) is designed as pre-requisites for a suitable implementation of human-like reasoning, and covers many of the issues mentioned above. It includes 20 sub-tasks taking the form of story reading and question answering. Sentences in each story describe some entities, with complex relations and interactions between them, based on a synthetic world. After several story sentences, a question about them is raised and requires a correct answer. Either 1k or 10k samples can be used for training. For each question, a subset of story sentences most relative to it are provided as "supporting facts". Several powerful models have been proposed in recent years to solve reasoning problems in bAbI (Graves et al., 2016; Kumar et al., 2016; Sukhbaatar, Weston, Fergus, et al., 2015). A common structure shared among these models is a writable external memory and soft attention addressing mechanism (Bahdanau, Cho, & Bengio, 2015). However, without external knowledge of logics, most of these models are data-thirsty. Up to now, the majority of good results reported on bAbI-20 are achieved by training on 10k, not 1k dataset. Moreover, end-to-end computing graphs in these models are not really compatible with the step-by-step characteristic of human reasoning due to lack of appropriate state-tracking mechanisms. Despite the synthetic nature of bAbI, some models cannot reach high accuracy on complex sub-tasks, such as path-finding (qa19) and positional-reasoning (qa17), even with 10k training samples.

There is close, intrinsic relationship between reasoning and human language. Therefore, partially re-constructing human logic system can be a straight-forward approach of reasoning. Research in cognitive linguistics (Johnson, 2013; Lakoff, 1989; Langacker, 1999) has shown that people's reactions to language can be described as Image Schema (IS). IS is a group of common cognitive patterns that appear repetitively in human brain while interacting with the environment. It is an abstraction of the physical world. For example, when people face the sentence "the book is on the table", they imagine that "there are two entities: *X* and *Y*, which are adjacent to each other but *X* (book) is on higher position and *Y* (table) is lower". The Image Schema activated by this sentence can be written as *(X, R, Y)*, in which *R* is an "abstract relation" having some specific logical properties. These logical properties are shared by a lot of real-world relations other than "is on", and thus critical for reasoning. For instance, the IS activated by phrases "is in" and

* Corresponding author at: Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China.
E-mail address: jiaming.xu@ia.ac.cn (J. Xu).

https://doi.org/10.1016/j.neunet.2018.06.012
0893-6080/

Table 1
An example of how logic rules are activated while reading real-world stories. *A* and *B* are a group of abstract relations having the logical properties of SPD-S (see Section 3). Our model learns to understand real-world relation "left" and activate the abstract relation *A*. With a successful variable binding, new relational tuples are generated, and more tuples are automatically inferred by the logic rules. These tuples are used to modify the tuple list in the working memory. Abstract relations catch common, essential logical properties, and thus are extensible to many other relational phrases in real-world texts.

Textual description	Variable binding	Abstract relations	IS activated	IS Inferred
The triangle is to the left of the square.	*1*:"triangle" *2*:"square"	For relations A and B:$(X, A, Y) \Leftrightarrow (Y, B, X)$; (X, A, Y) and $(Y, A, Z) \Rightarrow (X, A, Z)$.	$(1, A, 2)$	$(2, B, 1)$
The square is to the left of the star.	*3*:"star"	For relations J, K, …: (Other rules, dependent on the schema types.)	$(2, A, 3)$	$(3, B, 2)$ $(3, B, 1)$ $(1, A, 3)$

"belongs to" are the same with "is on" to some extent, in a sense that they both refer to an abstract relation that satisfy "if *(X, R, Y), (Y, R, Z)* then *(X, R, Z)*, but *(Z, not R, X)*". The same holds for a great amount of real-world relations. The number of basic abstract relations is limited, and they are the main components of Cognitive Model (CM) which is key to establishing concepts and performing reasoning. Although Image Schema (and the corresponding logic rules of abstract relations) can be formed by early experiences of the world, people must *learn* to activate the right schema while reading words or sentences to give the right response. Without this learning process, there is no *understanding* of language.

In this paper we propose a novel model using logic rules explicitly during reasoning. Our model maps natural language to sequences of *actions* modifying the whole IS in the *memory*. Logic rules serve as properties of abstract relations, and define how the IS changes when a new abstract relation is activated. Basic rules can be combined sequentially or recursively to solve complex tasks and reproduce human-like reasoning behavior. Our framework is different from a rule-based or knowledge-based system because our model learns to understand and make use of these rules from data, instead of directly applying them to reason by knowledge. Activating logic rules to boost reasoning performance involves at least two problems: (1) the same real-world entities in the story ("the book", "kitchen", etc.) must be mapped to the same abstract variables (*X*, *Y* s in relational tuples *(X, R, Y)*). (2) abstract relations (*R* s) must be activated to correctly change the tuple list representing the IS. We name the first problem *variable binding*, name of a human brain function which has not been well understood by neuroscience yet. The second problem is *relation activating*, which requires to map story sentences to a correct action sequence to change IS. Solving both (1) and (2) enables a model to track the state changes described by the textual stories, and generate an answer based on the final IS scene. Our model learns to solve (1) and (2) jointly under certain assumptions. Table 1 is an example of real-world story sentences with Image Schema, variable binding and abstract relations. We train our model via deep reinforcement learning with a reward signal generated according to whether the predicted answer is correct, and solve all 20 sub-tasks in bAbI-20 dataset nearly perfectly. Our model temporally uses an external system for Entity Recognition (ER), but does not rely on any kind of supporting facts, strong supervision or imitation learning.

Our contributions are three-folds:

- We propose a novel memory-augmented neural network framework making use of logic rules inducted from existing theories of Image Schema and Cognitive Model.[1]
- We redefine textual reasoning tasks as interactive-feedback process with human working memory. Under certain assumptions, we jointly solve two main problems: variable binding and relation activating, via deep reinforcement learning.
- Our experimental results show the existence of common properties among real-world relations, and the probability to partially re-construct human logic system to boost performances on language comprehension tasks. This can be a first step to better integration of rule-based and data-driven systems, and deeper understanding of the connections between machine learning and cognitive science/neuroscience/procedural knowledge.

The remainder of this paper is organized as follows: In Section 2, we review related works and give other perspectives to view our work. In Section 3, we introduce the details of our model. Section 4 illustrates experimental results and Section 5 proposes the conclusions.

2. Related work

The bAbI-20 dataset (Weston, Bordes et al., 2015), as a benchmark of textual reasoning, has drawn much attention. The 20 subtasks are all synthetic (without complicated sentences or large vocabulary) and provide pre-requisites for a complete intelligent agent, therefore suitable for basic research.

Memory Networks are a cluster of reasoning models that use external memory and learn to operate on it. MemNN (Weston, Chopra, & Bordes, 2015), MemN2N (Sukhbaatar et al., 2015) and Dynamic Memory Networks (Kumar et al., 2016; Xiong, Merity, & Socher, 2016) store the whole story as sentence embeddings in memory and learn to address with attentions. In these works, the majority of operations on memory is "reading". More complex methods include Neural Turing Machines (Graves, Wayne, & Danihelka, 2014), Dynamic Neural Turing Machines (Gulcehre, Chandar, Cho, & Bengio, 2016), Neural Random-Access Machines (Kurach, Andrychowicz, & Sutskever, 2016) and Differentiable Neural Computer (Graves et al., 2016), in which "writing to memory" operation also needs to be learned. These models are designed to use a sophisticated memory, and focus on basic experimental results on synthetic tasks. In our model, "Memory" is composed of a list of *(X, R, Y)* relational tuples representing the Image Schema (IS), and is modified by learned actions (binding entities and activating relations); "Reading" means to retrieve some entities or relations (or their continuous representations if necessary) from the memory; "Writing" means to append, modify or delete tuples in the list. Actions are taken after each sentence is read, without needing to see the whole story.

There are other important works on reasoning tasks. Our method is similar to Ent-Net (Henaff, Weston, Szlam, Bordes, & LeCun, 2017) in that both methods track the world states step-by-step instead of storing the whole story, and both models emphasis on the roles of entities. However, Ent-Net aims at a comparably general model and trains on large datasets such as bAbI-10k. In contrast, our work aims at precise reasoning, good interpretability of models, and perfect performances on small dataset (1k). Relation Networks (Santoro et al., 2017) propose a powerful module to enhance networks' reasoning ability of relational problems and solve

[1] Our code is available at https://github.com/FlamingHorizon/VRRM.

18/20 bAbI tasks with 10k samples, setting another strong baseline in a data-driven manner. The recently proposed Gated Graph Transformer Neural Network (Johnson, 2017) converts a textual story into a graphical structure via differentiable operations. The model handles most tasks on 1k dataset, if external supervision is added to each time step. On the other hand, our model deals with symbolic structures with discrete operations in a reinforcement manner, requiring more assumptions on the form of actions, but no supervision on the intermediate results. Lee et al. (2015) apply Tensor Product Representation to bAbI and analyze the common properties of relations. Our work also takes "common properties" as our standing point, but we require less external efforts in the semantic understanding process, and train word embeddings and action networks jointly.

Marblestone, Wayne, and Kording (2016) have discussed variable binding and provided some evidence of IS being acquired from early life experience, which is also our hypothesis in this work. In our works, logical properties of IS are stored as prior knowledge in long term memory, and described as logic rules between abstract relations. What needs to be learned is the mechanism to activate these rules while reading texts. Johnson (2013), Lakoff (1989), and Langacker (1999) established the theory of IS and CM and shed light on our work.

Promising results using the concepts of "schema" exists recently in the field of game playing with entity-based inputs (Kansky et al., 2017). Our method is different from the proposed Schema Networks in that we focus on raw natural language inputs, and explore the contribution of schema similarity to the performance of reasoning. Hu, Ma, Liu, Hovy, and Xing (2016) distill logic rules into network weights and boosts the performance in more realistic tasks. In contrast, our model learns to activate logic rules directly.

Deep Reinforcement Learning has been proved effective especially in text game tasks (Dhingra et al., 2017; Fatemi, El Asri, Schulz, He, & Suleman, 2016; He et al., 2015). Our work shows that the reasoning task can share some settings with them, i.e. modifying Image Schema based on textual input is a "text game" that can be modeled as sequential decision-making process.

Ahn, Choi, Pärnamaa, and Bengio (2016) proposed to use relational tuples as external knowledge in language modeling tasks. Their knowledge base contains relational tuples as real-world facts, while our "knowledge base" mainly includes logic rules defining how (dynamically generated) relational tuples interact with each other. Their model belongs to deep learning, and encodes relational tuples into vectors and take part in neural network computations, while we generate symbolic relational tuples to take part in external reasoning process.

3. Model

We redefine textual reasoning as a sequential decision-making problem, and solve it with reinforcement learning (RL). The elements of RL are listed below:

State: A list of relational tuples in the working memory (as environment), and the story sentences (as input).

State Transition: The insertion, deletion or modification of the tuple list, based on actions and logic rules.

Action: Generating new tuples (or none) used to modify or retrieve from the tuple list. Typical actions can be decomposed into variable binding and abstract relation activating.

Reward: Depends on whether the question is correctly answered.

Our model structure is composed of a symbolic working memory (a tuple list), an input module, an action network and an answer generation module. We first show details of each module, and then demonstrate how they are combined to solve the reasoning tasks. We name our model the *Variable-Relation Reasoning Machine (VRRM)*.

3.1. Abstract relations and logic rules

The memory used in our model stores the whole scene of Image Schema. We represent it as a list of *(X, R, Y)* tuples, containing variables and their relations. The abstract relations (*R*s) are distinguished by the different logic rules. In this paper we mainly use the following three kinds of abstract relations, corresponding to three of the six kinds of basic Kinesthetic Image Schema described in Lakoff (1989):

3.1.1. Part-whole schema

Part-Whole Schema (PW-S) models a wide range of simple real-world relations such as "someone is at somewhere". They cannot interact with other relations or have other logic properties. If an abstract relation *R* belongs to PW-S, we just append *(X, R, Y)* into the tuple list while it is activated.

3.1.2. Source–Path–Destination schema

The Source–Path–Destination Schema (SPD-S) is another critical intuition for precise reasoning. It models human sense of spatial relations (i.e. below–above-left–right, or east–west-north–south). For abstract relations in SPD-S, the most important properties are opposition and transitivity. Many real-world relations other than spatial relations can also be modeled as SPD-S. See Section 4.1.1 for more examples.

Abstract relations belonging to SPD-S always appear in pairs or groups, satisfying the following three rules:

- In each group, the two relations, named *A* and *B*, are opposite to each other: If *(X, A, Y)*, then *(Y, B, X)*. We write it as $A = B^{-1}$ and vice versa.
- Every single relation itself is transitive: If *(X, A, Y)* and *(Y, A, Z)*, then *(X, A, Z)*.
- Relations in different groups (i.e. {*A, B* }, {*C, D* }) can combine to form new relations. If *(X, A, Y)* is activated and *(Y, C, Z)* is already in the working memory, then *(X, AC, Z)* and *(Z, DB, X)* is also added. But there is no further rules for *AC* and *DB* for simplicity, except opposition.

As an example of SPD-S, the real-world relations {"east of", "west of"} may activate abstract relations {*A, B*}, and {"south of", "north of"} may activate {*C, D*}. Then *AC* and *DB* are two new relations modeling "southeast" and "northwest". There is no further rules for *AC* and *DB* except $AC = (DB)^{-1}$.

3.1.3. Link schema

Link Schema (L-S) defines three relations *A*, *B* and *E*:

- If *(X, A, Y)* and *(Z, A, Y)* then *(X, E, Z)* and *(Z, E, X)*.
- If *(X, E, Z)* and *(Z, B, W)* then *(X, B, W)*.

E often indicates some kinds of "equivalence". Once two entities are linked by *E*, they share some other properties defined by *B*. This schema is important for induction. For instance, *A* ⇔ "belongs to"; *Y* ⇔ "bird species"; *E* ⇔ "in same species"; *B* ⇔ "has ability"; *W* ⇔ "fly".

3.2. Input module

In every time step *t*, the model takes one sentence as input. A sentence with *n* one-hot words $\{x_{1(t)}, x_{2(t)}, \ldots, x_{n(t)}\}$ is first embedded by a word vector table *A*:

$$e_{i(t)} = Ax_{i(t)}. \tag{1}$$

Word vectors are combined to a sentence embedding by an encoder:

$$s_t = encoder\left(e_{1(t)},\ e_{2(t)}, \ldots\right). \tag{2}$$

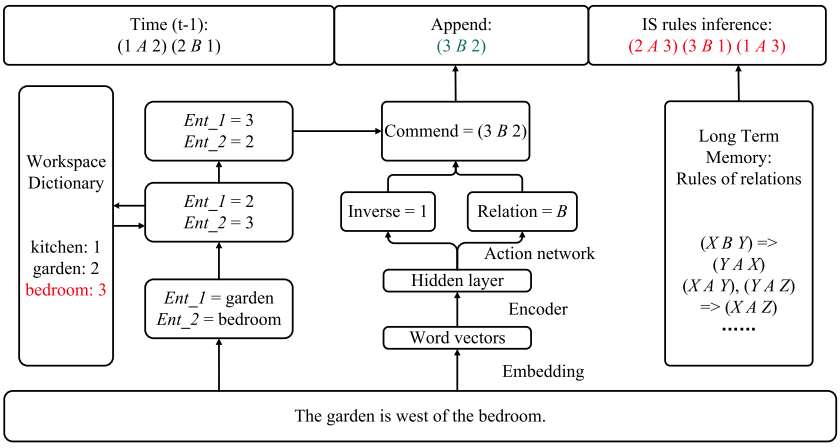

Fig. 1. Behavior of the action network in one time step. *A* and *B* are a pair of abstract relations that satisfies the rules of SPD-S. The agent must learn to set up link between real world relation "west of" and one of the SPD-S relations *A* or *B*. The working memory stores previous tuples, the command generated by action network, and new tuples inferred according to the logic rules. Items newly added to the variable dictionary or newly inferred by logic rules in this step are colored red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In our experiments, we use the concatenation of all word vectors as the sentence encoder. The length of s_t is fixed to $L * D$, where L is the maximum sentence length and D is the dimension of word embeddings. Sentences not in the full length are padded by zeros. There could be many other forms of encoders, such as LSTM (Hochreiter & Schmidhuber, 1997), or taking sum according to word position, like Sukhbaatar et al. (2015), etc.

The question q is embedded in the same way as story sentences, sharing the embedding weights.

3.3. Action network

While reading a story sentence, our model generates new relational tuples to modify the tuple list representing the working memory; while reading a question, our model generates tuples as queries to retrieve from the final working memory. Action network generates actions to control the generation of all these tuples. It takes the sentence embedding s_t from Input Module as input, and outputs a distribution over feasible actions.

To generate new tuples, we first recognize the entities in a sentence and map them to certain IDs using a key–value variable dictionary like the "Variable Binding" column in Table 1. For Entity Recognition, we just do POS-tagging and select by hand since the entity-set in bAbI is relatively small. These steps are automatically done and not learned. Then, for each pair of entities {ID1, ID2} in the sentence, a parameterized action network predicts an abstract relation between them, and decides the role of each entity as subject or object. The behavior of the action network in one step is shown in Fig. 1, see also Table 3. We define two kinds of actions:

1. **Activate**: There is a candidate abstract relation set Relat = {A, B, C, …} with each relation belonging to a certain Schema. The action network selects a proper relation R to activate.
2. **Bind**: Decide the role of each entity, binding it to the subject or object. There is a candidate order set Ord = {0,1}. If order == 0, a tuple (ID1, R, ID2) will be generated; else the reversed tuple (ID2, R, ID1) will be generated.

For each entity pair, the overall candidate action set is Relat × Ord. There is an additional candidate action "Remove" that deletes all tuples containing these 2 entities (this corresponds to the human brain function of filtering out-dated information). If there are more than 2 entities in a sentence, we take Cartesian product for all entity pairs. For example, if a sentence x_t is "Daniel passed the apple to Sandra", there are three pairs of entities: (Daniel, Sandra), (apple, Sandra) and (Daniel, apple). For each pair, there can be two abstract relation candidates {A, B} and two order candidates {0, 1}. Then the action space is of size $(2 * 2 + 1)^3 = 125$. Most sentences in bAbI-20 do not contain more than 3 entities. We further reduce computational costs by only considering the direct neighborhoods of each entity in the sentence. Also, if abstract relations {A, B} have "opposition" properties (like SPD-S), the "Bind" actions can be redundant and ignored. Thus, in our experiments, the action space in each time step is usually much smaller (<10).

We use Multi-Layer Perception with one hidden layer and sigmoid (σ) nonlinearity as our action network. The number of neurons in the output layer equals to the number of action candidates.

The output of action network is a sequence of distributions of valid actions:

$$\pi(b_t = b_t^j | s_t) = softmax(W_{bind}\sigma(W_{hid}s_t), \tau), \tag{3}$$

$$\pi(a_t = a_t^j | s_t) = softmax(W_{activate}\sigma(W_{hid}s_t), \tau). \tag{4}$$

At each time t (T sentences in total), b_t and a_t represent the variable binding and relation activating action to choose, respectively; b_t^j and a_t^j stand for the jth feasible candidate for b_t and a_t; W_{bind}, $W_{activate}$ and W_{hid} are the network connection weights; τ is the temperature constant of softmax distribution, which is a hyperparameter. The joint distribution of b_t and a_t is then written as follows:

$$\pi^t_{action}(b_t, a_t | s_t) = \pi(b_t | s_t)\pi(a_t | s_t), \tag{5}$$

under the assumption of conditional independency. We use this assumption because a_t usually depends on the predicates and prepositions in sentence s_t, while b_t is often related to the sentence voice.

To process the question, an action includes deciding both relation and order in the same way as above. We write it as r_i for simplicity. For the ith question q_i (N questions in total):

$$\pi^i_{retr}(r_i = r_i^k | q_i) = softmax(W_{retr}\sigma(W_q q_i), \tau), \tag{6}$$

where r_i represents the action to choose for the ith question; r_i^k is the kth feasible candidate for r_i.

While reading a question, the order and abstract relations decided by the action network are combined to generate new tuples used to form queries to retrieve from the tuple list. Different types of queries are specialized for different tasks. These types are hand-designed and fixed as behaviors of environment. See Table 2 for examples.

Table 2
Examples for questions, outputs of action networks, and queries. "Guess" means that in the answer module, R will be compared to another result $R^{'}$, to answer yes/no. $R^{'}$ is achieved by retrieving from the tuple list with a query like ID1, ?, ID2. This is specialized only for yes/no tasks in bAbI-20.

Task question type	Action outputs	Query type
What color is X?	Relation R; order	(ID, R, ?)
How do you go from X to Y?	Order	(ID1, ?, ID2)
Is X west of Y?	Relation R; order	(ID1, ?, ID2); Guess R

Retrieving from the tuple list with a query means finding the first tuple in the list that matches all the existing part of the query, and returns the value of the missing part ("?"). If there is no match, return a special token "no_result". This is automatically done by the environment, and action network only learns to generate the correct query.

The joint probability of the whole action sequence $a_{1...T}$, $b_{1...T}$, $r_{1...N}$ is

$$\pi(a_{1...T},\ b_{1...T}, retr_{1...N} \mid s_{1...T}, q_{1...N}) = \prod_{t=1}^{T} \pi^{t}_{act}(\mathrm{b}_t, \mathrm{a}_t \mid s_t) \prod_{i=1}^{N} \pi^{i}_{retr}(r_i \mid q_i)\ . \tag{7}$$

3.4. Answer module

Answer module converts the retrieved results to answer words. Answer words can come from the story text, or be generated independently. In most cases, answer words can be achieved by just looking up the variable dictionary (ID-to-string) inversely. In other tasks, the agent must read the retrieved results and choose a word from all possible answers. In this case, the answer module needs to be parameterized to perform an extra step in decision-making sequence. For example, when the question is "Is the kitchen west of the garden?", the query generated by action network may be *(id_kitchen, ?, id_garden)*; using it to retrieve from the tuple list yields a result R' (perhaps corresponds to real-world relation "east" or others); then the action network also guesses an abstract relation R (perhaps corresponds to the word "west" in the question). The parameterized answer module reads R and R' and generates an answer word "no". The answer word distribution is computed with:

$$\pi^{i}_{ans}(w_i = w^{m}_{i} | e_i) = softmax(W_{ans}\sigma(W_e e_i), \tau), \tag{8}$$

where w_i is the answer word to choose for question i, and w^{m}_{i} stands for the mth feasible answer word; e_i is a vector encoding the retrieve results and other information (like the guessed answer). For e_i we use a one-hot representation of corresponding entities or relations, and find it effective enough for all tasks.

Table 3 shows how all the modules introduced above are combined to solve reasoning problems.

3.5. Training

In our settings, there is no supervision on the actions, thus no human behavior to imitate directly. This is a suitable case for reinforcement learning. We generate a reward signal by comparing the predicted answer word with the true label, and supervise the whole action sequence with this single reward. We apply the REINFORCE algorithm (Williams, 1992), a widely used policy gradient method, to minimize the objective function:

$$L(\theta) = E\left[\sum_{h=0}^{H} \gamma^{h} r_h\right]. \tag{9}$$

Table 3
Behaviors of VRRM in the whole reasoning process. The colored steps are done by parameterized modules; others are automatically completed by the environment. In step 13 and 15, if (ID1, R1, ID2) is generated but (ID1, R2, ID2) already exists in the tuple list, the later one is removed because of conflict.

Algorithm: the whole reasoning process of VRRM

Inputs: story sentences $x_{1...T}$, questions $q_{1...N}$.
Begin with: tuple list = []; variable dictionary = {}.
1: **for** each x_t:
2: Encode x_t into vector s_t using Input Module .
3: Recognize all entities with external Entity Recognition methods.
4: Register an ID (1,2, ...) for each entity with variable dictionary.
5: Action Network reads s_t and:
6: **for** each two entity IDs (ID1, ID2):
7: Choose relation R from abstract relation candidates {A,B, ... }.
8: Decide the order of ID1 and ID2.
9: **if** R == "Remove":
10: Remove all tuples containing ID1 and ID2 in the tuple list.
11: **else**:
12: Generate tuple tu = (ID1, R, ID2)/(ID2, R, ID1) for order == 0/1.
13: Append tu into tuple list; remove all tuples in conflict with tu.
14: According to the Schema of R, activate logic rules to infer new tuples.
15: Append all inferred tuples into tuple list, and remove conflicts.
16: **end if**
17: Final tuple list is achieved after reading all sentences.
18: **for** each question q_t:
19: Run step 2-4 on q_t. Achieve entity ID3 (or more, ID4, ID5, ...)
20: Action Network reads the encoded q_t and:
21: *## An example for questions like "what color is Greg".*
22: Choose a relation $R^{'}$ from abstract relation candidates {A,B, ... }.
23: Decide the order (ID3 being subject or object in the query).
24: Generate a query like (ID3, $R^{'}$, ?).
25: Retrieve the final tuple list with the query and find ID_ANS to fill the missing part.
26: *## An example of the simplest form of Answer Module:*
27: Look up from the inverse variable dictionary with ID_ANS and generate the corresponding word as answer. (can be parameterized.)
Outputs: the answer of each question.

Gradients are computed with:

$$\nabla L(\theta) = E_{a\sim\pi}\left[\left(\sum_{h=0}^{H} \nabla_\theta \log \pi_\theta\ (a_h)\right)\sum_{h=0}^{H} \gamma^{h} r_h\right]. \tag{10}$$

In each episode (simulation), we sample the whole action a_h sequence of variable binding, relation activating, memory retrieving and answer word generation from the policy distribution π; r_h is the immediate reward received at time step h, and γ is the time decay constant of reward. We estimate the expectation by executing multiple simulations. The gradient updates are done in mini-batches with RMSProp (Tieleman & Hinton, 2012), and the learning rate is set to $1e-4$ in most of our experiments.

4. Experiments

4.1. Single task results

Table 4 shows our test error rates on all 20 sub-tasks using 1k training set. Our model performs at the same level with state-of-the-arts using 10k for training, and better than other models using 1k dataset. It also has much less parameters and performs steadily, while data-driven methods on 10k dataset usually need multi-runs to select the best one.

4.1.1. Tasks and schemas

The Source–Path–Destination Schema (SPD-S) is a critical intuition for precise reasoning. When the number of "relation groups" is 2, it models human sense of the 2-dimensional space (i.e. below–above-left–right, or east–west-north–south). At the same time, the properties of SPD-S relations can model lots of other real-world relations such as "belong to". The first 9 subtasks in Table 4

Table 4
Test error rates (%) on bAbI-1k dataset. Some important results achieved by well-known models on 1k/10k dataset are listed on the right side. RN (Santoro et al., 2017) refers to "Relation Networks", and the original paper did not give all error rates and claimed to have solved 18/20 tasks (<5% error rates in all tasks except qa2 and qa3). GGT-NN (Johnson, 2017) proposed two versions: weakly-supervised and strong-supervised. We compare to the strong-supervised one, which uses hand-designed graphs and performs much better than the weakly-supervised one.

Tasks & Results	Schema	1k			10k			
		VRRM	GGTNN (Johnson, 2017)	MemN2N (Sukhbaatar et al., 2015)	EntNet (Henaff et al., 2017)	DMN+ (Xiong et al., 2016)	DNC (Graves et al., 2016)	RN (Santoro et al., 2017)
2 two-supporting-facts	SPD-S	0.0	0.0	11.4	0.1	0.3	0.4	>5
3 three-supporting-facts	SPD-S	0.0	1.3	2.9	4.1	1.1	1.8	>5
4 two-arg-relations	SPD-S	3.5	1.2	13.4	0.0	0.0	0.0	–
7 counting	SPD-S	0.0	0.0	18.3	0.0	2.4	0.6	–
8 lists-sets	SPD-S	0.0	0.0	9.3	0.5	0.0	0.3	–
15 basic-deduction	SPD-S	0.0	0.9	0.0	0.0	0.0	0.0	–
17 positional-reasoning	SPD-S	3.1	34.5	40.4	0.5	4.2	12.0	–
18 size-reasoning	SPD-S	0.3	2.1	9.2	0.3	2.1	0.8	–
19 path-finding	SPD-S	0.0	0.0	88.0	2.3	0.0	3.9	–
1 single-supporting-fact	PW-S	0.0	0.0	0.0	0.0	0.0	0.0	–
5 three-arg-relations	PW-S	1.1	1.6	14.4	0.3	0.5	0.8	–
6 yes–no-questions	PW-S	0.0	0.0	10.1	0.2	0.0	0.0	–
9 simple-negation	PW-S	0.0	0.0	1.5	0.1	0.0	0.2	–
10 indefinite-knowledge	PW-S	0.8	3.4	6.5	0.2	0.0	0.0	–
11 basic-coreference	PW-S	0.0	0.0	0.3	0.3	0.0	0.0	–
12 conjunction	PW-S	0.0	0.1	0.0	0.0	0.2	0.0	–
13 compound-coreference	PW-S	0.0	0.0	0.2	0.3	0.0	0.0	–
20 agents-motivations	PW-S	0.0	0.0	0.0	0.0	0.0	0.0	–
14 time-reasoning	TIM	4.7	2.2	6.9	0.0	0.2	0.4	–
16 basic-induction	L-S	0.5	0.0	2.7	0.2	45.3	55.1	–

correspond to a wide range of real-world relations: directions, belongings, pass-to actions, and even animal natures such as "be afraid of", but they are in fact similar in the concept of Image Schema (IS) and can be activated as same abstract relations in SPD-S. The Part-Whole Schema (PW-S) models simplest real-world relations and can be used in combination with SPD-S. PW-S and SPD-S are compatible in formulas because with correct variable binding, the rule of transitivity does not influence the learning process much in tasks irrelevant to SPD-S. We use PW-S for the rest of the 20 tasks except qa14 and 16.

Qa14 and 16 cannot be solved well with knowledge of SPD-S and PW-S. In task 14 (time-reasoning), the logic rules between relations are closely related to the concept of time (TIM). The rules of "time" are specially written for qa14. The agent needs to understand each sentence and handle real-world relations such as "be somewhere in the morning/afternoon/evening". If person X goes to places A, B and C in sequence, *(X, A, B), (X, B, C)* are appended in the tuple list. Link Schema (LS) is used for "basic induction" problem (qa16), and it is essential to the formation of semantic boundaries and concepts.

4.1.2. Training details

In tasks like qa1, 12 and 16, since the entity in a question is unique (notified as *X*), our model predicts a relation *R* and forms a query *(X, R, ?)* to retrieve from the final working memory. The answer word is generated by mapping the retrieved ID-number back into word string using the variable dictionary. In qa19, the retrieve action is to decide the order of the two entities in the question, and use query *(X, ?, Y)* or *(Y, ?, X)* to find a relation as result. We use the parameterized version of answer module to generate answer words that do not exist in story texts, (i.e."s, e"[2]) and this increases the difficulty of decision-making. In yes/no tasks such as qa6, 9, 17 and 18, our model guesses a result based on the question only, and compares it with the retrieve result. A parameterized answer module takes one-hot representation of comparison information as input, and chooses an answer word from "yes" or "no".

We set dimension of word embeddings and number of hidden-layer units to 20 in all tasks. Since there are often lots of sub-optimal solutions in these tasks, the temperature constant τ affects the convergency and final error rates as an important balance factor between exploratory and exploitation. For each task, we search τ from {1, 10, 20} and pick a best result. The reward is set to {−0.1 for wrong answer and 1 for correct} for yes–no tasks, and {−1 for wrong and 5 for correct} for others. Only the final reward achieved by the answer word is used in our experiments, and the time decay constant γ is set to 1.

In each epoch, we take 200 simulations for each question, resulting in a pool of 200,000 training trails, and perform 5 steps of gradient descent in batch-size 32. In all tasks, our model converges within 80 epochs.

4.1.3. What does the model learn?

We attribute the superior performance to the logic rules behind Image Schema and variable dictionary to help binding variables and acquiring concepts. We illustrate the scalability and similarity to human behavior, which are unique for an IS based reasoning model.

Table 5 shows the ability of our model to decide variable orders. The sequence is taken from qa19 results. The agent is able to learn to activate *A*, *B*, *C* and *D* while the sentence describes real-world relations "east", "west", "south" and "north", respectively. The model understands the opposition between the positions and learns to re-produce the same binding sequence as human. The decision of reversing the entity order for question (which is in accordance with human behavior) reveals the power of the model to distinguish between different sentence structures and tones.

Table 6 shows an action sequence from qa16 using Link Schema. The model comprehends every sentence, decides the order of entities (all must be positive order), and predicts the relation to be *A* or *B* in the L-S defined in Section 3. Note that the activated abstract relations are based on the whole sentence instead of solely on the predicates, because all the predicates in the story are "is", but the meanings are different.

4.2. Cross category training

If multiple groups of abstract relations are available, the action learned from data may not necessary be all the same with humans,

[2] "s, e" means "go south, go east" in the answer word set of qa19.

Table 5
The whole action sequence produced by our module trained on qa19 using Source–Path–Destination Schema. For each sentence, order = 0 means *(1, B, 2)* and order = 1 means *(2, B, 1)*, etc.

Story	Variable	Order	Action
The garden is west of the bathroom.	1:garden		
	2:bathroom	0	1 *B* 2
The bedroom is north of the hallway.	3:bedroom		
	4:hallway	0	3 *D* 4
The office is south of the hallway.	5:office	0	5 *C* 4
The bathroom is north of the bedroom.	–	0	2 *D* 3
The kitchen is east of the bedroom.	6:kitchen	0	6 *A* 3
How do you go from the bathroom to the hallway?	–	1	4 ? 2

Table 6
The whole action sequence produced by our model trained on qa16 using Linking Schema.

Story	Action
Lily is a frog.	1 *A* 2
Bernhard is a frog.	3 *A* 2
Bernhard is green.	3 *B* 4
Brian is a lion.	5 *A* 6
Brian is white.	5 *B* 7
Julius is a swan.	8 *A* 9
Julius is green.	8 *B* 4
Lily is green.	1 *B* 4
Greg is a swan.	10 *A* 9
What color is Greg?	10 *B* ?

Table 7
The whole action sequence produced by our model trained on qa6 using multiple categories of Schema.

Story	Action
Sandra journeyed to the bathroom.	1 *A* 2
Sandra got the milk there.	3 *C* 2
Daniel traveled to the kitchen.	4 *A* 5
Sandra traveled to the office.	1 *A* 6
Daniel journeyed to the office.	4 *A* 6
Sandra moved to the bathroom.	1 *A* 2
Is Sandra in the bathroom?	Retrieve: (1 2 ?) Guess: *A*

if there are "better" choices. We test on qa6 on this setting, asking the model to make choice among two PW-S relations *A*, *B*, and a group of SPD-S relations {*C, D* }. The model gets 0.0% error rates, even better than results achieved without SPD-S. Table 7 shows the whole sequence. Note that abstract relation *C* belongs to SPD-S, but the model activates it to represent a PW-S relation ("got milk") without influencing the final results, because the transitivity property of SPD-S can be ignored while not interacting with other abstract relations.

A more difficult setting is joint training on tasks including various kinds of real-world relations. We generate a new dataset including the scenarios in both qa-16 (L-S), qa-19 (SPD-S, direction), qa-2 (SPD-S, belongings) and qa-20 (PW-S, people feelings), but limit the story length to 4. Each story can include all kinds of these real-world relations, and the agent must search in a large space (more than 7 candidate abstract relations at each time step) to decide the correct action sequence. We got 3.4% error rate on this dataset, showing large capability to handle complex scenarios.

4.3. Generalization to longer sequences

Generalization is a characteristic of human reasoning: a rule that holds must hold for any instance. For extremely long stories, it is difficult or time-consuming to train a deep reinforcement model with 16 or more action choices at each step. Nevertheless, once the model learns to use the logic rules, it can generalize to arbitrary story length and preserve the right decision. To demonstrate the efficiency of our model on long stories, we train on qa18, limit the story length to 5, and test the model on the full-length test set, in which the largest length of the story can be up to 15. The ratio of stories longer than 5 in the dataset is 20.7%, and we get 0.3% error rate on this setting, indicating that the reasoning procedure can generalize to long sequences.

4.4. Train-sample efficiency

The core idea of our model is learning to activate rules. A very small training set of bAbI may be enough for relatively simple rules. For qa6, we train on {50, 100, 200, 400, 1000} training samples, and get {15.3%, 15.0%, 9.0%, 0.1%, 0.0%} test error rates, showing that 400 samples leads to a nearly perfect result. We attribute this to both the synthetic nature of bAbI and the ability of our model to quickly acquire real-world concepts and map them to abstract logic expressions. During reinforcement learning, the multiple simulations on each single sample also help increase sample efficiency.

5. Conclusions

In this paper, we propose a novel neural reasoning model augmented by a discrete external memory. It mimics the Image Schema (IS) in human cognitive activities and tracks the world states on-the-fly. IS enables our model to learn to activate logic rules explicitly. The model jointly learns variable binding and relation activating to solve all sub-tasks in bAbI-20 with less than 1k training examples. Future work includes combining a few schemas to solve real-world language processing tasks, or adding in more (yet limited number of) schemas to re-construct more important parts of human logic systems, using hierarchical and recursive methods.

Acknowledgments

We thank the reviewers for their insightful comments, and this work was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB02070005), the National Natural Science Foundation (Grant No. 61602479) and the Independent Deployment Project of CAS Center for Excellence in Brain Science and Intelligent Technology (Grant No. CEBSIT2017-02).

References

Ahn, S., Choi, H., Pärnamaa, T., & Bengio, Y. (2016). A neural knowledge language model. arXiv preprint arXiv:1608.00318.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., et al. (2017). End-to-end reinforcement learning of dialogue agents for information access. In *ACL*.

Fatemi, M., El Asri, L., Schulz, H., He, J., & Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. In *17th annual meeting of the special interest group on discourse and dialogue* (p. 101).

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. arXiv preprint arXiv:1410.5401.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471–476.

Gulcehre, C., Chandar, S., Cho, K., & Bengio, Y. (2016). Dynamic neural turing machine with soft and hard addressing schemes. arXiv preprint arXiv:1607.00036.

He, J., Chen, J., He, X., Gao, J., Li, L., & Deng, L. et al., (2015). Deep reinforcement learning with a natural language action space. arXiv preprint arXiv:1511.04636.

Henaff, M., Weston, J., Szlam, A., Bordes, A., & LeCun, Y. (2017). Tracking the world state with recurrent entity networks. In *ICLR*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hu, Z., Ma, X., Liu, Z., Hovy, E., & Xing, E. (2016). Harnessing deep neural networks with logic rules. In *ACL*.

Johnson, M. (2013). *The body in the mind: The bodily basis of meaning, imagination, and reason*. University of Chicago Press.

Johnson, D. D. (2017). Learning graphical state transitions. In *ICLR*.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., & Lou, X. et al., (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. arXiv preprint arXiv:1706.04317.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., et al. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *ICLR*.

Kurach, K., Andrychowicz, M., & Sutskever, I. (2016). Neural random-access machines. In *ICLR*.

Lakoff, G. (1989). *Women, fire, and dangerous things: What categories reveal about the mind*. University of Chicago Press.

Langacker, R. W. (1999). *Grammar and conceptualization, Vol. 14*. Walter de Gruyter.

Lee, M., He, X., Yih, W.-t., Gao, J., Deng, L., & Smolensky, P. (2015). Reasoning in vector space: An exploratory study of question answering. arXiv preprint arXiv:1511.06426.

Marblestone, A. H., Wayne, G., & Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, *10*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., et al. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems* (pp. 4974–4983).

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *NIPS*.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*(2).

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., & Joulin, A. et al., (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arxiv:1502.05698.

Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. In *ICLR*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256.

Xiong, C., Merity, S., & Socher, R. (2016). Dynamic memory networks for visual and textual question answering. In *ICLR*.