ReasoNet: Learning to Stop Reading in Machine Comprehension

Yelong Shen, Po-Sen Huang, Jianfeng Gao, Weizhu Chen
Microsoft Research Redmond
{yeshen, pshuang, jfgao, wzchen} @ microsoft.com

Abstract

Teaching a computer to read a document and answer general questions pertaining to the document is a challenging yet unsolved problem. In this paper, we describe a novel neural network architecture called Reasoning Network (ReasoNet) for machine comprehension tasks. ReasoNet makes use of multiple turns to effectively exploit and then reason over the relation among queries, documents, and answers. Different from previous approaches using a fixed number of turns during inference, ReasoNet introduces a termination state to relax this constraint on the reasoning depth. With the use of reinforcement learning, ReasoNet can dynamically determine whether to continue the comprehension process after digesting intermediate results, or to terminate reading when it concludes that existing information is adequate to produce an answer. ReasoNet has achieved state-of-the-art performance in machine comprehension datasets, including unstructured CNN and Daily Mail datasets, and a structured Graph Reachability dataset.

1 Introduction

Teaching machines to read, process, and comprehend natural language documents is a coveted goal for artificial intelligence [2, 17, 6]. Genuine reading comprehension is extremely challenging, since effective comprehension involves thorough understanding of documents and performing sophisticated inference. Toward solving this machine reading comprehension problem, in recent years, several work has collected various datasets, in the form of question, passage, and answer, to test machine on answering a question based on the provided passage [17, 6, 7, 16]. Some large-scale cloze-style datasets [6, 7] have gained significant attention along with powerful deep learning models.

Recent approaches on cloze-style datasets can be separated into two categories: single-turn and multi-turn reasoning. Single turn reasoning models utilize attention mechanisms [1] with deep learning models to emphasize specific parts of the document which are relevant to the query. These attention models subsequently calculate the relevance between a query and the corresponding weighted representations of document subunits (e.g. sentences or words) to score target candidates [7, 6, 8]. However, considering the sophistication of the problem, after a single-turn comprehension, readers often revisit some specific passage or the question to grasp a better understanding of the problem. With this motivation, recent advances in reading comprehension have made use of multiple turns to infer the relation between query, document and answer [7, 5, 22, 18]. By repeatedly processing the document and question after digesting intermediate information, multi-turn reasoning can generally produce a better answer and all existing work has demonstrated its superior performance consistently.

Existing multi-turn models have a fixed number of hops or iterations in their inference, i.e., with predetermined reasoning depth, without regard to the complexity of each individual query or document. However, when a human reads a document with a question in mind, we often decide whether we want to stop reading if we believe the observed information is adequate already to answer the question, or continue reading after digesting intermediate information until we can answer the question with confidence. This behavior generally varies from document to document, or question to question

because it is related to the sophistication of the document or the difficulty of the question. Meanwhile, the analysis in [3] also illustrates the huge variations in the difficulty level with respect to questions in the CNN/Daily Mail datasets [6]. For a significant part of the datasets, this analysis shows that the problem cannot be solved without appropriate reasoning on both its query and document.

With this motivation, we propose a novel neural network architecture called Reasoning Network (ReasoNet). ReasoNet tries to mimic the inference process of human readers. With a question in mind, ReasoNet reads a document repeatedly, each time focusing on different parts of the document until a satisfied answer is found or formed. This reminds us of a Chinese proverb: *"The meaning of a book will become clear if you read it hundreds of times."*. Moreover, unlike previous approaches using fixed number of hops or iterations, ReasoNet introduces a termination state in the inference. This state can decide whether to continue the inference to next turn after digesting intermediate information, or to terminate the whole inference when it concludes that existing information is sufficient to yield an answer. This number of turns in the inference is dynamically modeled by both the document and the query, and will be learned automatically according to the difficulty of the problem.

One of the significant challenges ReasoNet faces is how to design an efficient training method, since the termination state is discrete and not connected to the final output. This prohibits canonical back-propagation method being directly applied to train ReasoNet. Inspired by [24, 13], we tackle this challenge by proposing a novel deep reinforcement learning method called Contrastive Reward (CR) to successfully train ReasoNet. Unlike traditional reinforcement learning optimization methods using a global variable to capture rewards, CR utilizes an instance-based reward baseline assignment. Experiments show the superiority of CR in both training speed and accuracy. Finally, by accounting for a dynamic termination state during inference and applying proposed deep reinforcement learning optimization method, ReasoNet is able to achieve the state-of-the-art results in machine comprehension datasets, including unstructured CNN and Daily Mail datasets, and a proposed structured Graph Reachability dataset.

This paper is organized as follows. In section 2, we review and compare recent work on machine reading comprehension tasks. In section 3, we introduce our proposed ReasoNet model architecture and training objectives. Section 4 presents the experimental setting and results on unstructured and structured machine reading comprehension tasks .

2 Related Work

Recently, with large-scale datasets available and the impressive advance of various statistical models, machine reading comprehension tasks have attracted much attention. Here we mainly focus on the related work in cloze-style datasets [6, 7]. Based on how they perform the inference, we can classify their models into two categories: single-turn and multi-turn reasoning.

Single-turn reasoning Single turn reasoning models utilize an attention mechanism to emphasis some sections of a document which are relevant to a query. This can be thought of as treating some parts unimportant while focusing on other important ones to find the most probable answer. Hermann et al. [6] propose the attentive reader and the impatient reader models using neural networks with an attention over passages to predict candidates. Hill et al. [7] use attention over window-based memory, which encodes a window of words around entity candidates, by leveraging an end-to-end memory network [19]. Meanwhile, given the same entity candidate could appear multiple times in a passage, Kadlec et al. [8] propose the attention-sum reader to sum up all the attention scores for the same entity. This score captures the relevance between a query and a candidate. Chen et al. [3] propose using a bilinear term similarity function to calculate attention scores with pretrained word embedding. Trischler et al. [22] propose the EpiReader which uses two neural network structures: one extracts candidates using attention-sum reader; the other reranks candidates based on a bilinear term similarity score calculated from query and passage representations.

Multi-turn reasoning For complex passages and complex queries, human readers often revisit the given document in order to perform deeper inference after reading a document. Several recent studies try to simulate this revisit by combining the information in the query with the new information digested from previous iterations [7, 5, 18, 23, 12]. Hill et al. [7] use multiple hops memory network to augment the query with new information from the previous hop. Gated Attention reader [5] is an extension of the attention-sum reader with multiple iterations by pushing the query encoding into an attention-based gate in each iteration. Iterative Alternative (IA) reader [18] produces a new

Algorithm 1: Stochastic Inference in ReasoNet

Input : Memory M; Initial state s_1; Step $t = 1$; Maximum Step T_{max}
Output : Termination Step T, Answer a_T

Generate binary termination variable $t_t \sim p(\cdot|f_t(s_t; \theta_t))$;
if t_t is false, go to Step 3; otherwise Step 6;
Generate attention vector $x_t = f_{att}(s_t, M; \theta_x)$;
Update internal state $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$;
Set $t = t + 1$; if $t < T_{\text{max}}$ go to Step 1; otherwise Step 6;
Generate answer $a_t \sim p(\cdot|f_a(s_t; \theta_a))$;
Return $T = t$ and $a_T = a_t$;

query glimpse and document glimpse in each iteration and utilizes them alternatively in the next iteration. Cui et al. [4] further propose to extend the query-specific attention to both query-to-document attention and document-to-query attention, which is built from the intermediate results in the query-specific attention. By reading documents and enriching the query in an iterative fashion, multi-turn reasoning has demonstrated their superior performance consistently.

Our proposed approach explores the idea of using both attention-sum to aggregate candidate attention scores and multiple turns to attain a better reasoning capability. Unlike previous approaches using fixed number of hops or iterations, motivated by [14, 13], we propose a termination module in the inference. The termination module can decide whether to continue to infer the next turn after digesting intermediate information, or to terminate the whole inference process when it concludes existing information is sufficient to yield an answer. The number of turns in the inference is dynamically modeled by both a document and a query, and is generally related to the complexity of a document and the difficulty of a query.

3 Reasoning Networks

ReasoNet is devised to mimic the inference process of human readers. ReasoNet reads a document repeatedly, with attention on different parts each time until a satisfied answer is found. As shown in Figure 1, ReasoNet is composed of the following components:

Internal State: The internal state is denoted as S which is a vector representation of the question state. Typically, the initial state s_1 is the last-word vector representation of query by an RNN. The t-th time step of the internal state is represented by s_t. The sequence of internal state is modeled by an RNN: $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$;

Memory: The external memory is denoted as M. It is a list of word vectors, $M = \{m_i\}_{i=1..D}$, where m_i is a fixed dimensional vector. In machine comprehensive tasks, m_i is the vector representation of each word in the doc by a bidirectional-RNN.

Attention: Attention vector x_t is generated based on the current internal state s_t and the external memory M: $x_t = f_{att}(s_t, M; \theta_x)$;

Termination Gate: Termination gate will produce a stochastic random variable according to the current internal state; $t_t \sim p(\cdot|f_t(s_t; \theta_t)))$. t_t is a binary random variable. If t_t is true, the ReasoNet will stop, and the answer module will execute at the t time step; otherwise the ReasoNet will generate an attention vector x_{t+1}, and feed into the state network to update the next internal state s_{t+1}.

Answer: The action of answer module will be triggered when the termination gate variable is true: $a_t \sim p(\cdot|f_a(s_t; \theta_a))$.

In Algorithm 1, we describe the stochastic inference process of the ReasoNet. The process could be considered as a Partially Observable Markov Decision Process (POMDP) [9] in the reinforcement learning (RL) literature. The state sequence $s_{1:T}$ is hidden and dynamic, controlled by an RNN sequence model. ReasoNet performs an answer action a_T at the T-th step, which implies that the termination gate variables $t_{1:T} = (t_1 = 0, t_2 = 0, ..., t_{T-1} = 0, t_T = 1)$. ReasoNet learns a stochastic policy $\pi((t_t, a_t)|s_t; \theta)$ with parameters θ to get a distribution over termination actions, to continue reading or to stop, and over answer actions if the model decides to stop at the current step. The termination step T varies from instance to instance.

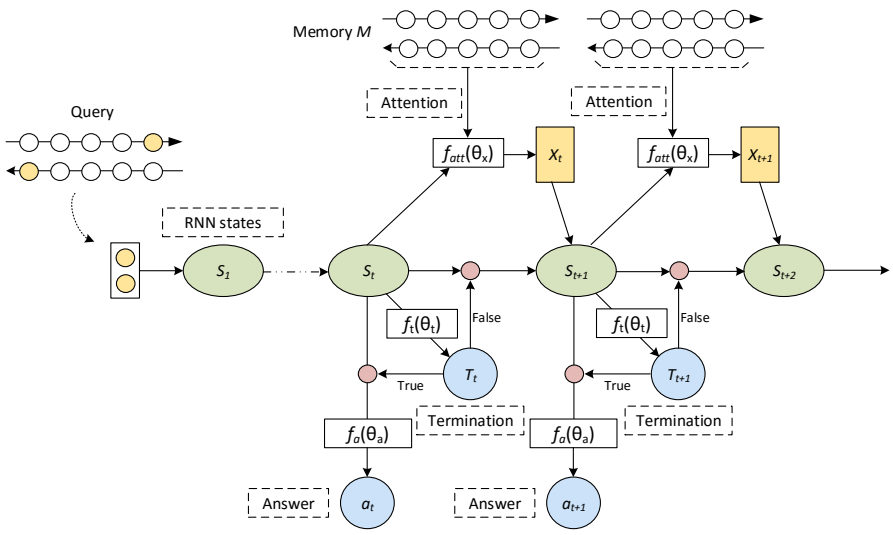


Figure 1: ReasoNet Architecture.

The parameters of the ReasoNet θ are given by the parameters of the attention network θ_x, the state RNN network θ_s, the answer action network θ_a, and the termination gate network θ_t. The parameters $\theta = \{\theta_x, \theta_s, \theta_a, \theta_t\}$ are trained by maximizing the total reward that the ReasoNet could expect when interacting with the environment. The expected reward for an instance is defined as:

$$J(\theta) = \mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\sum_{t=1}^{T} r_t \right]$$

The reward can only be received at the final termination step when an answer action a_T is performed. We define $r_T = 1$ if $t_T = 1$ and the answer is correct, and $r_T = 0$ otherwise. The rewards on intermediate steps are zeros, $\{r_t = 0\}_{t=1...T-1}$. J can be maximized by directly applying gradient based optimization methods. The derivative of J on θ is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\nabla_\theta \log \pi(t_{1:T}, a_T; \theta) r_T \right]$$

In practice, it is usually to apply the REINFORCE algorithm [24] to estimate $\nabla_\theta J(\theta)$:

$$\mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\nabla_\theta \log \pi(t_{1:T}, a_T; \theta) r_T \right] = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) \left[\nabla_\theta \log \pi(t_{1:T}, a_T; \theta)(r_T - b_T) \right]$$

where $\mathbb{A}^\dagger$ is all the possible episodes, $T, t_{1:T}, a_T$ and r_T are the termination step, termination action, answer action, and reward, respectively, for the $(t_{1:T}, a_T)$ episode. b_T is called the reward baseline in the RL literature to lower variance [21]. It is common to select $b_T = \mathbb{E}_\pi [r_T]$ [20], and can be updated via an online moving average approach : $b_T = \lambda b_T + (1 - \lambda) r_T$.

However, we empirically find that above approach leads to slow convergence in training ReasoNet. Intuitively, the average baselines $\{b_T; T = 1..T_{\max}\}$ are global variables independent of instances. It is hard for them to capture the dynamic termination behavior of ResoNet. In other words, ReasoNet may stop at different time steps for different instances. The adoption of a global variable without considering the dynamic variance in each instance is inappropriate. To resolve this weakness in traditional methods and account for the dynamic characteristic of ReasoNet, we propose an instance-based baseline method called "Contrastive Reward" (CR) to calculate $\nabla_\theta J(\theta)$. The basic idea of CR is to utilize an instance-based baseline assignment. We will elaborate its implementation details in Section 3.1. Empirical results show that the proposed reward schema has produced better results compared to the baseline approach.

3.1 Training Details

In the machine reading comprehension tasks, a training dataset can be simplified as a collection of triplets of query $\mathbf{q}$, passage $\mathbf{p}$, and answer $\mathbf{a}$. Say $\langle q_n, p_n, a_n \rangle$ is the n-th training instance.

The first step is to extract memory M from p_n by mapping each symbolic in the passage to a contextual representation given by the concatenation of forward and backward RNN hidden states, i.e., $m_k = [\overrightarrow{p_n}^k, \overleftarrow{p_n}^{|p_n|-k+1}]$, and extract initial state s_1 from q_n by assigning $s_1 = [\overrightarrow{q_n}^{|q_n|}, \overleftarrow{q_n}^1]$. Given M and s_1 for the n-th training instance, ReasoNet would run with $|\mathbb{A}^\dagger|$ episodes, where all possible episodes $\mathbb{A}^\dagger$ can be enumerated by setting a maximum step. Each episode generates actions and a reward from the last step: $\langle (t_{1:T}, a_T), r_T \rangle_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger}$.

Therefore, the derivative of J on θ can be rewritten as:

$$\nabla_\theta J(\theta) = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) \left[\nabla_\theta \log \pi(t_{1:T}, a_T; \theta)(r_T - b) \right]$$

where the baseline $b = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) r_T$ is the average reward on the $|\mathbb{A}^\dagger|$ episodes for the n-th training instance. It allows different baselines for different training instances. This can be beneficial since the complexity of training instances varies significantly. Since the sum of the proposed rewards over $|\mathbb{A}^\dagger|$ episodes is zero, $\sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta)(r_T - b) = 0$, we call it Contrastive Reward in this work. In experiments, we empirically find using $(\frac{r_T}{b} - 1)$ in replace of $(r_T - b)$ could lead to a faster convergence. Therefore, we adopt this approach to train the ReasoNet in the experiments.

4 Experiments

4.1 CNN and Daily Mail Datasets

We evaluate the performance of ReasoNet on CNN and Daily Mail datasets.[1] Table 1 provides some statistics on the two datasets. The detailed settings of the ReasoNet model are as follows.

Vocab Size: For training our ReasoNet, we keep the most frequent $|V| = 101k$ words (not including 584 entities and 1 placeholder marker) in the CNN dataset, and $|V| = 151k$ words (not including 530 entities and 1 placeholder marker) in the Daily Mail dataset.

Embedding Layer: We choose word embedding size $d = 300$, and use the 300 dimensional pretrained Glove word embeddings [15] for initialization. We also apply dropout with probability 0.2 to the embedding layer.

Bi-GRU Encoder: We apply bi-directional GRU for encoding query and passage into vector representations. We set the number of hidden units to be 256 and 384 for the CNN and Daily Mail datasets, respectively. The recurrent weights of GRUs are initialized with random orthogonal matrices. The other weights in GRU cell are initialized from a uniform distribution between -0.01 and 0.01. We use a shared GRU model for both query and passage.

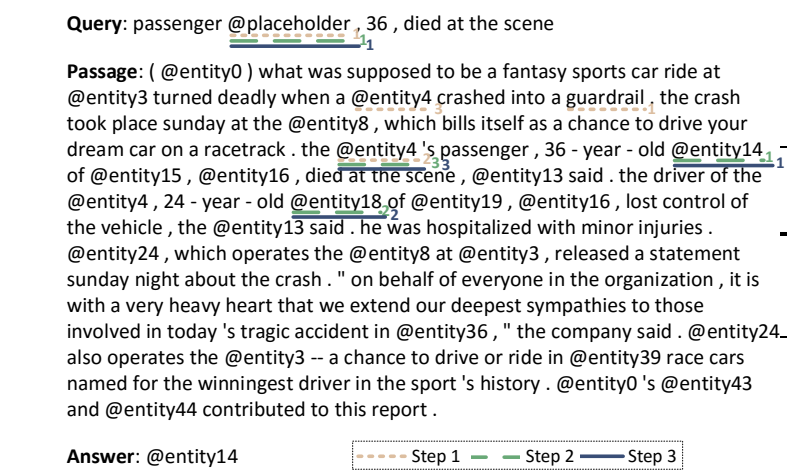
Memory and Attention: The memory of ReasoNet on CNN and dailymail dataset is composed of query memory and passage memory. $M = (M^{query}, M^{doc})$, where M^{query} and M^{doc} are extracted from query bidirectional-GRU encoder and passage bidirectional-GRU encoder respectively. We choose projected cosine similarity function as the attention module. The attention score $a^{doc}_{t,i}$ on memory m^{doc}_i given state s_t is computed as follows: $a^{doc}_{t,i} = \text{softmax}_{i=1,\ldots,|M^{doc}|} \gamma \cos(\mathbf{w_1^{doc}} m^{doc}_i, \mathbf{w_2^{doc}} s_t)$, where γ is set to 10. $\mathbf{w_1^{doc}}$ and $\mathbf{w_2^{doc}}$ are weight vectors associated with m^{doc}_i and s_t, respectively, and are joint trained in the ReasoNet. Thus, attention vector on passage is given by $x^{doc}_t = \sum_i^{|M|} a_{t,i} m^{doc}_i$. The final attention vector is the concatenate of the query attention vector and the passage attention vector $x_t = (x^{query}_t, x^{doc}_t)$. The attention module is parameterized by $\theta_x = (\mathbf{w_1^{query}}, \mathbf{w_2^{query}}, \mathbf{w_1^{doc}}, \mathbf{w_2^{doc}})$;

Internal State Controller: We choose GRU model as the internal state controller. The number of hidden units in the GRU state controller is 256 for CNN and 384 for Daily Mail. The initial state

[1] The CNN and Daily Mail datasets are available at https://github.com/deepmind/rc-data

Table 1: Statistics of CNN news data and Daily Mail news reading comprehension datasets.

	CNN News			Daily Mail News		
	train	valid	test	train	valid	test
# Query	380,298	3.924	3,198	879,450	64,835	53,182
Max # candidates	527	187	396	371	232	245
Avg # candidates	26.4	26.5	24.5	26.5	25.5	26.0
Avg # tokens	762	763	716	813	774	780
Vocab size		118,497			208,045	



Step	Termination Probability	Attention Sum
1	0.0011	0.4916
2	0.5747	0.5486
3	0.9178	0.5577

Figure 2: Results of a test example 69e1f777e41bf67d5a22b7c69ae76f0ae873cf43.story from the CNN dataset. The numbers next to the underline bars indicate the rank of the attention scores. The corresponding termination probability and the sum of attention scores for the answer entity are shown in the table on the right.

of the GRU controller is set to be the last-word of the query representation by a bidirectional-GRU encoder.

Termination Module: We adopt a logistical regression to model the termination variable at each time step : $f_t(s_t; \theta_t) = \text{sigmoid}(\mathbf{w_t} s_t + \mathbf{b_t}); \theta_t = (\mathbf{w_t}, \mathbf{b_t})$

Answer Module: We apply a linear projection from GRU outputs and make predictions on the entity candidates. Following the settings in AS Reader [8], we sum up scores from the same candidate and make a prediction. Thus, AS Reader can be viewed as a special case of ReasoNet with $T_{\max} = 1$.

Other Details: The maximum reasoning step, $T_{\max}$ is set to 5 in experiments on both CNN and Daily Mail Datasets. We use ADAM optimizer [10] for parameter optimization with an initial learning rate of 0.0005, $\beta_1 = 0.9$ and $\beta_2 = 0.999$; The absolute value of gradient on each parameter is clipped within 0.001. The batchsize is 64 for both CNN and Daily Mail datasets. For each batch of the CNN and Daily Mail datasets we randomly reshuffle the assignment of named entities [6]. This forces the model to treat the named entities as semantically meaningless labels. In the prediction of test cases, we randomly reshuffle named entities up to 4 times, and report the averaged answer. Models are trained on GTX TitanX 12GB. It takes 7 hours per epoch to train on the Daily Mail dataset and 3 hours per epoch to train on the CNN dataset. The models are usually converged within 6 epochs on both CNN and Daily Mail datasets.

Table 2 shows the performance of all the existing single model baselines and our proposed ReasoNet . By capturing multi-turn reasoning and learning to stop reading a paragraph, we have achieved the state-of-the-art results in both CNN and Daily Mail datasets. To further understand the inference process of ReasoNet, Figure 2 shows a test example of the CNN dataset. The model initially focuses on wrong entities with low termination probability. In the second and third steps, the model focuses on the right clue with higher termination probability. Interestingly, we also find that query attention focuses on the placeholder token throughout all the steps.

Table 2: The performance of Reasoning Network on CNN and Daily Mail dataset.

	CNN		Daily Mail	
	valid	test	valid	test
Deep LSTM Reader [6]	55.0	57.0	63.3	62.2
Attentive Reader [6]	61.6	63.0	70.5	69.0
MemNets [7]	63.4	66.8	-	-
AS Reader [8]	68.6	69.5	75.0	73.9
Stanford AR [3]	72.2	72.4	76.9	75.8
DER Network [11]	71.3	72.9	-	-
Iterative Attention Reader [18]	72.6	73.3	-	-
EpiReader [22]	**73.4**	74.0	-	-
GA Reader [5]	73.0	73.8	76.7	75.7
AoA Reader [4]	73.1	74.4	-	-
ReasoNet	72.9	**74.7**	**77.6**	**76.6**

Table 3: Reachability statistics of the Graph Reachability dataset.

	Small Graph				Large Graph			
Reachable Step	No Reach	1–3	4–6	7–9	No Reach	1–3	4–6	7–13
Train (%)	44.16	42.06	13.51	0.27	49.02	25.57	21.92	3.49
Test (%)	45.00	41.35	13.44	0.21	49.27	25.46	21.74	3.53

4.2 Graph Reachability Task

Recent analysis and results [3] on the cloze-style machine comprehension tasks have suggested some simple models without multi-turn reasoning can achieve reasonable performance. Based on these results, we construct a synthetic structured Graph Reachability dataset[2] to evaluate longer range machine inference and reasoning capability, since we expect ReasoNet has the capability to handle long range relationships.

We generate two synthetic datasets: a small graph dataset and a large graph dataset. In the small graph dataset, it contains $500K$ small graphs, where each graph contains 9 nodes, and 16 direct edges to randomly connect pairs of nodes. The large graph dataset contains $500K$ graphs, where each graph contains 18 nodes, and 32 random direct edges. Duplicated edges are removed. Table 3 shows the graph reachability statistics on the two datasets.

In Table 4, we show examples of a small graph and a large graph in the synthetic dataset. Both graph and query are represented by a sequence of symbols. In the experiment, we use a 100-dimensional embedding vector for each symbol, and bidirectional-LSTM with 128 and 256 cells for query and graph embedding in the small and the large graph datasets, respectively. The last states of bidirectional-LSTM on query are concatenated to be the initial internal state $s_1 = [\overrightarrow{q}^{|q|}, \overleftarrow{q}^{1}]$ in the ReasoNet. Another bidirectional-LSTM on graph description maps each symbol g^i to a contextual representation given by the concatenation of forward and backward LSTM hidden states $m_i = [\overrightarrow{g}^{i}, \overleftarrow{g}^{|g|-i+1}]$. The final answer is "Yes" or "No" and hence logistical regression is used as the answer module: $a_t = \sigma(\mathbf{w_a} s_t + \mathbf{b_a})$; we applied another logistical regression as the termination gate module: $t_t = \sigma(\mathbf{w_t} s_t + \mathbf{b_t})$. The maximum reasoning step $T_{\max}$ is set to 15 and 25 for the small graph and large graph dataset, respectively.

We study the effectiveness of the termination gate in ReasoNet. We denote "ReasoNet" as the standard ReasoNet with termination gate, as described in Section 3.1. If we remove the termination gate, and just simply use the last state answer action as the final answer, say $\hat{a} = a_{T_{\max}}$ ($T_{\max}$ is the maximum reasoning step), denoted as "ReasoNet-Last". To study the effectiveness of multi-turn reasoning, we choose "ReasoNet-$T_{\max} = 2$", which only has single-turn reasoning, as a baseline.

In Table 5, we report the performance of ReasoNet, ReasoNet-Last and ReasoNet-$T_{\max} = 2$ models on the Graph Reachability dataset. The ReasoNet-Last model performs very well on the small graph dataset, and it could obtain 100% accuracy. However, the ReasoNet-Last model fails to learn on the large graph dataset, as the task becomes much more challenging. Meanwhile, the ReasoNet model

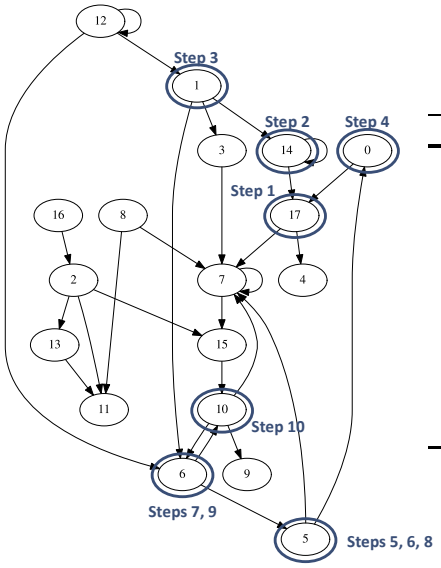
[2]The dataset is available at https://github.com/MSRDL/graph_reachability_dataset

Table 4: Small and large random graph in the Graph Reachability dataset. Note that "A –> B" represents an edge connected from A to B and the # symbol is used as a delimiter between different edges.

	Small Graph	Large Graph
Graph Description	0 –> 0 # 0 –> 2 # 1 –> 2 # 2 –> 1 # 3 –> 2 # 3 –> 3 # 3 –> 6 # 3 –> 7 # 4 –> 0 # 4 –> 1 # 4 –> 4 # 5 –> 7 # 6 –> 0 # 6 –> 1 # 7 –> 0 #	0 –> 17 # 1 –> 3 # 1 –> 14 # 1 –> 6 # 2 –> 11 # 2 –> 13 # 2 –> 15 # 3 –> 7# 5 –> 0 # 5 –> 7 # 6 –> 10 # 6 –> 5# 7 –> 15 # 7 –> 7 # 8 –> 11 # 8 –> 7 # 10 –> 9 # 10 –> 6 # 10 –> 7 # 12 –> 1 # 12 –> 12 # 12 –> 6 # 13 –> 11 # 14 –> 17 # 14 –> 14 # 15 –> 10 # 16 –> 2 # 17 –> 4 # 17 -> 7 #
Query	7 –> 4	10 –> 17
Answer	No	Yes

Table 5: The performance of Reasoning Network on the Graph Reachability dataset.

	Small Graph			Large Graph		
	ROC-AUC	PR-AUC	Accuracy	ROC-AUC	PR-AUC	Accuracy
ReasoNet-$T_{\max} = 2$	0.9638	0.9677	0.8961	0.8477	0.8388	0.7607
ReasoNet-Last	**1**	**1**	**1**	0.8836	0.8742	0.7895
ReasoNet	**1**	**1**	**1**	**0.9988**	**0.9989**	**0.9821**

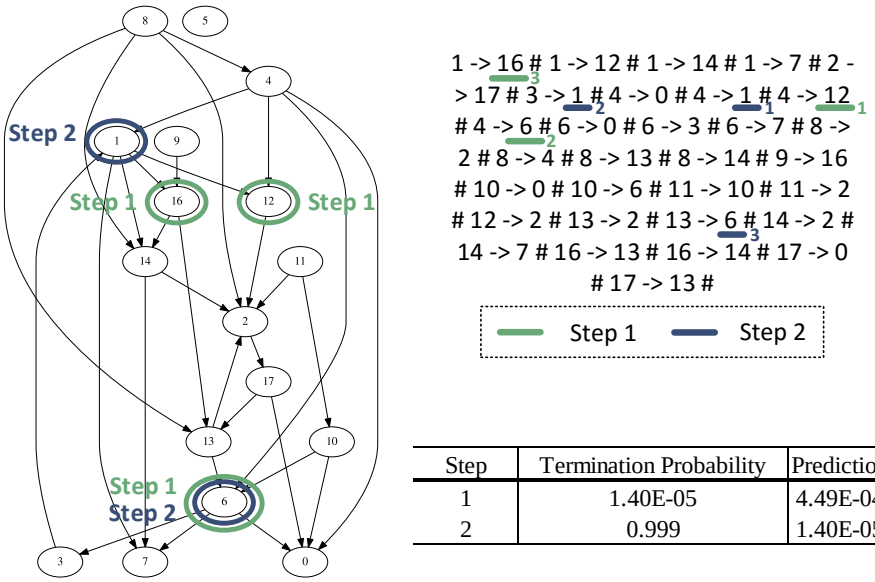


Step	Termination Probability	Prediction
1	1.00E-06	0.172
2	1.00E-06	0.625
3	1.00E-06	0.752
4	1.00E-06	0.202
5	1.00E-06	0.065
6	1.00E-06	0.041
7	2.30E-06	0.137
8	0.0017	0.136
9	0.49	0.761
10	0.99	0.927

Figure 3: An example of graph reachability result, given a query "10 –> 17" (Answer: Yes). The red circles highlight the nodes/edges which have the highest attention in each step. The corresponding termination probability and prediction results are shown in the table. The model terminates at step 10.

converges faster than the ReasoNet-Last model. The ReasoNet model converges in 20 epochs in the small graph dataset, and 40 epochs in the large graph dataset, while the ReasoNet-Last model converges around 40 epochs in the small graph dataset, and 70 epochs in the large graph dataset. The results suggest that the termination gate variable in ReasoNet is helpful when training with sophisticated examples, and makes models converge faster. Both the ReasoNet and ReasoNet-Last models perform better than the ReasoNet-$T_{\max} = 2$ model, which demonstrates the importance of multi-turn reasoning.

To further understand the inference process in the ReasoNet, Figures 3 and 4 show test examples of the large graph dataset. In Figure 3, we can observe that the model does not make a firm prediction till



Step	Termination Probability	Prediction
1	1.40E-05	4.49E-04
2	0.999	1.40E-05

Figure 4: An example of graph reachability result, given a query "4 –> 9" (Answer: No). The numbers next to the underline bars indicate the rank of the attention scores. The corresponding termination probability and prediction results are shown in the table.

step 9. The highest attention word at each step shows the reasoning process of the model. Interestingly, the model starts from the end node (17), traverses backward till finding the starting node (10) in step 9, and makes a firm termination prediction. On the other hand, in Figure 4, the model learns to stop in step 2. In step 1, the model looks for neighbor nodes (12, 6, 16) to 4 and 9. Then, the model gives up in step 2 and predict "No". All of these demonstrate the dynamic termination characteristic and potential reasoning capability of ReasoNet.

5 Conclusion

In this paper, we propose a ReasoNet that dynamically decides whether to continue or to terminate the inference process in machine comprehension tasks. Using reinforcement learning with the proposed contractive reward, our proposed model has shown to achieve the start-of-the-art results in machine comprehension datasets, including unstructured CNN and Daily Mail datasets, and a proposed structured Graph Reachability dataset. For future work, ReasoNet can be generalized to other tasks that requires reasoning capability, such as question answering and knowledge graph inference.

Acknowledgments

We thank Ming-Wei Chang, Li Deng, Lihong Li and Xiaodong Liu for their thoughtful feedback and discussions.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015.

[2] Léon Bottou. From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149, 2014.

[3] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the CNN / Daily Mail reading comprehension task. In *ACL*, 2016.

[4] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *CoRR*, abs/1607.04423, 2016.

[5] Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *CoRR*, abs/1606.01549, 2016.

[6] Karm Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701, 2015.

[7] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. the Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*, 2016.

[8] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv:1603.01547v1 [cs.CL]*, 2016.

[9] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[11] Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies (NAACL-HLT)*, 2016.

[12] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*, 2016.

[13] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[14] Rodrigo Nogueira and Kyunghyun Cho. Webnav: A new large-scale task for natural language based sequential decision making. In *Advances in Neural Information Processing Systems*, 2016.

[15] Richard Socher Pennington, Jeffrey and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 2014.

[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[17] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 2013.

[18] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 2016.

[19] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[20] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

[21] Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, 1984.

[22] Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. *CoRR*, abs/1606.02270, 2016.

[23] Dirk Weissenborn. Separating answers from queries for neural reading comprehension. 2016.

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