FINDING REMO (RELATED MEMORY OBJECT): A SIMPLE NEURAL ARCHITECTURE FOR TEXT BASED REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory Network based models have shown a remarkable progress on the task of relational reasoning. Recently, a simpler yet powerful neural network module called Relation Network (RN) has been introduced. Despite its architectural simplicity, the time complexity of relation network grows quadratically with data, hence limiting its application to tasks with a large-scaled memory. We introduce *Related Memory Network*, an end-to-end neural network architecture exploiting both memory network and relation network structures. We follow memory network's four components while each component operates similar to the relation network without taking a pair of objects. As a result, our model is as simple as RN but the computational complexity is reduced to linear time. It achieves the state-of-the-art results in jointly trained bAbI-10k story-based question answering and bAbI dialog dataset.

1 INTRODUCTION

Neural network has made an enormous progress on the two major challenges in artificial intelligence: seeing and reading. In both areas, embedding methods have served as the main vehicle to process and analyze text and image data for solving classification problems. As for the task of logical reasoning, however, more complex and careful handling of features is called for. A reasoning task requires the machine answer a simple question upon the delivery of a series of sequential information. For example, imagine that the machine is given the following three questions: "Mary got the milk there.", "John moved to the bedroom.", and "Mary traveled to the hallway." Once prompted with the question, "Where is the milk?", the machine then needs to sequentially focus on the two sentences, "Mary got the milk there" and "Mary traveled to the hallway" in order to successfully determine that the milk is located in the hallway.

Inspired by this reasoning mechanism, J. Weston & Bordes (2015) has introduced the memory network (MemNNs), which consists of an external memory and four components: input feature map (I), generalization (G), output feature map (O), and response (R). The external memory enables the model to deal with a knowledge base without loss of information. Input feature map embeds the incoming sentences; Generalization updates old memories given the new input; Output feature map finds relevant information from the memory, and; Response produces the final output. End-to-end memory networks (MemN2N) (Sukhbaatar et al., 2015), gated end-to-end memory networks (GMemN2N) (Liu & Perez, 2017) , dynamic memory networks (DMN) (Kumar et al., 2016), and dynamic memory network + (DMN+) (Xiong et al., 2016) are all based on memory network architecture.

Recently, a new architecture called Relation Networks (RNs) (Santoro et al., 2017) has been proposed as a general solution to relational reasoning. While DMN, DMN+, GMemN2N has to go through complicated computation process, RN simply learns from the compositions of two multilayer perceptrons (MLPs) yet achieve better performance as compared to the previous three models. RN's simplicity origins from its unique input form which is a set of objects. The design philosophy behind RNs is to to directly capture the relation between the object pair. However, since it considers a pair at a time, the computation complexity becomes quadratic. Quadratic complexity limits the number of pairs of objects that the model can handle, hence making it difficult for the model to be

augmented with large scale external memory. Santoro et al. (2017) has suggested attention mechanisms as a solution to filter out unimportant relations; however, since it interrupts the reasoning operation, it may not be the most optimal solution to the problem.

We propose "Related Memory Networks" (RMNs) as an alternative mechanism to effectively solve the reasoning task. The architecture is inspired by the close connection between memory network and relation network. RMN is as simple as RN, yet its computational complexity is reduced to $O(n)$. It consists of an external memory and four components: embedding, attention, updating, and reasoning. Each component is composed of a MLP, making RMN end-to-end differentiable. Experiment results show that RMN achieves state-of-the-art accuracy on bAbI story-based question answering and bAbI dialog dataset.

2 Related Work

2.1 Memory-Augmented Neural Network

To answer the question from a given set of facts, the model needs to memorize these facts from the past. Long short term memory (LSTM) (Hochreiter & Schmidhuber, 1997), one of the variants of recurrent neural network (RNN), is inept at remembering past stories because of their small internal memory (Sukhbaatar et al., 2015). To cope with this problem, J. Weston & Bordes (2015) has proposed a new class of memory-augmented model called Memory Network (MemNN). MemNN comprises of an external memory m and four components: input feature map (I), generalization (G), output feature map (O), and response (R). I encodes the input x to $I(x)$ and G updates the memory given the $I(x)$ ($m = G(m, I(x))$), whereas O reads output feature o from the memory ($o = O(I(x), m)$). Finally, R infers an answer from o.

MemN2N, GMemN2N, DMN, and DMN+ closely resembles MemNN's structure, in which each component is a neural network that can be trained end-to-end. Differences between these models comes from the way the components are constructed.

MemN2N consists of I, O, and R components without G. O first computes output feature o based on the inner product of the embedded facts and the prompt question. R produces an answer given the sum of the output vector o and the embedded question. GMemN2N exhibits a similar structure as MemN2N, except that R yields an answer from the gated sum of the o and the embedded question. DMN and DMN+ are made up of all four components–I, G, O, and R–where G and O are modelled in a more delicate manner. They calculate o from the concatenation of a variety of similarities between input (c), memory (m) and question (q) vectors, such as $c \circ q$, $c \circ m$, $|c - q|$, $|c - m|$ and so on. G updates memory with gated recurrent units (GRUs) (Chung et al., 2014).

Performance of these models are sensitive to how finely each component is constructed. in constrast, RMN uses simpler MLPs so model sensitivity is reduced, while showing better performance.

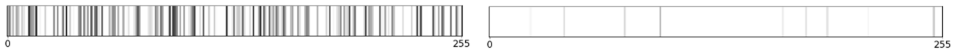
2.2 Relation Network

Unlike memory-augmented neural network models which select and update necessary information in a complex and sophisticated manner, Relation Networks (RN) have emerged as a new and simpler framework for solving the inference problem. Under the assumption that relations are inferred from the connection between an object and another, RN takes in a pair of objects and the prompt question as its input, of which the mathematical expressions is as following:

$$RN(O) = f_\phi(\sum_{i,j} g_\theta(o_i, o_j, q)) \tag{1}$$

where g_θ and f_ϕ are MLPs with parameters θ and ϕ. The output of g_θ is relation, whether two objects are related in or not, and f_ϕ aggregates all these outputs to infer the answer.

RN has an advantage of solving the task through a MLP without any other complicated processes, but it requires $O(n^2)$ complexity for computation because the model has to concatenate two objects together as a pair. Santoro et al. (2017) has pointed out that that prior knowledge could be reflected at this point and suggested employing attentional mechanisms in order to filter out unimportant



(a) related objects (b) unrelated objects

Figure 1: The output vector of g_θ (RN) when the input objects are related and unrelated

relations. However, incorporating attention before the construction of the object pair weakens model performance. For example, suppose that we compute an attention weight α from a given attention module. The weighted sum of the α_i to each o_i, regardless of the quality of the attention model employed, blurs the relation between the set of objects.

3 MODEL

3.1 OUTPUT FEATURE MAP AND g_θ

We found a close connection between output feature map (O) and g_θ used in RN. O component retrieves the output vector through the attention weight between 0 and 1. It indicates how relevant is the story to the question and can be calculated via cosine similarity or concatenation of many interactions between them. g_θ also finds meaningful relation with the given question. Different from O, this simple MLP directly yields the output vector implying whether the input objects are related or not, as seen in figure 1.

Motivated by the potential of MLP to focus on the related object, we use MLP to produce the attention weight without any extrinsic computation between the input sentence and the question. By virtue of little modification, RMN was able to be a simple but strong model with reduced time complexity.

3.2 RELATED MEMORY NETWORK

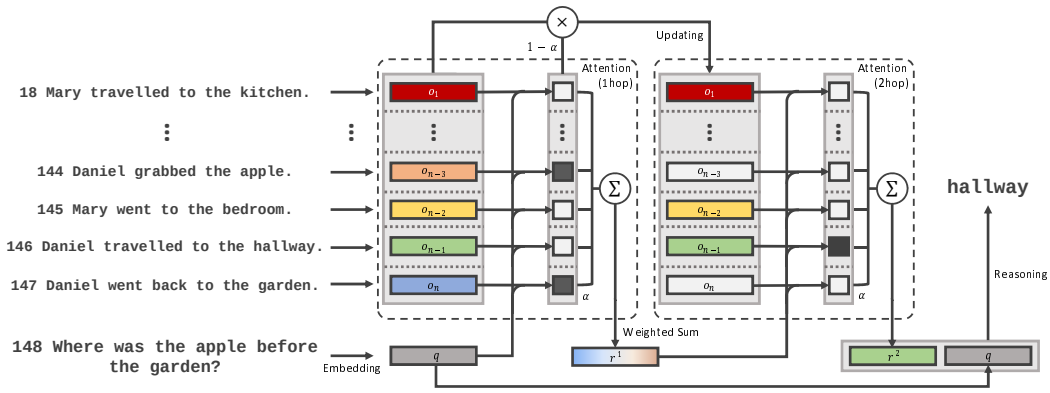


Figure 2: Related Memory Network

Our model, Related Memory Network (RMN), is composed of four components - embedding, attention, updating, and reasoning. It takes the inputs a set of stories $x_1, x_2, ..., x_n$ and its related question q, and outputs an answer a. Each of the x_i, q, and a is made up of one-hot representation of words, for example, $x_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{in_i}\}$ $(x_{ij} \in \mathbb{R}^V, j = (1, 2, ..., n_i), V =$ vocabulary size, $n_i =$ number of words in sentence $i)$. Stories are stored in the memory and flow to the following components.

Embedding component We first embed words in each $x_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{in_i}\}$ and q to a continuous space multiplying an embedding matrix $A \in \mathbb{R}^{d \times V}$. Then, the sentence x_i can be represented to an object o_i, using any of the following five methods: simple sum (equation 2), position

encoding (J. Weston & Bordes, 2015) (equation 3), concatenation (equation 4), LSTM, and GRU. In case of LSTM or GRU, o_i is the final hidden state of it.

$$o_i = \sum_j Ax_{ij} \tag{2}$$

$$o_i = \sum_j l_j \cdot Ax_{ij} \quad (l_{kj} = (1 - j/n_i) - (k/d)(1 - 2j/n_i)) \tag{3}$$

$$o_i = [Ax_{i1}; Ax_{i2}; ... ; Ax_{in_i}] \tag{4}$$

As the following attention component takes the concatenation of o_i and q, it is not necessarily that story and question to have the same dimensional embedding vectors unlike previous memory-augmented neural networks.

Attention component Attention component can be applied more than once depending on the problem; figure 2 illustrates 2 hop version of RMN. We refer to the ith embedded sentence on the tth hop as o_i^t.

Previously mentioned in section 3.1, we applied simple MLPs represented as g_θ^ts to constitute the attention component. g_θ^t must be ended with 1 unit output layer to provide a scalar weight w_i^t, which leads to an attention weight α_i^t. In the beginning, a vector concatenated with o_i^1 and q flows to the g_θ^1. From the result of g_θ^1, attention weight α_i^1 is calculated using additional variable β^1 to control the intensity of attention, inspired by how Neural Turing Machine (Graves et al., 2014) reads from the memory. Then we get the related memory object r_i^1, a weighted sum of α_i^1 and object o_i^1. If there exist more hops, r_i^1 is taken to the next hop and iterates over this process with the updated memory object o_i^2. All the procedures are rewritten as equation 5, 6, and 7:

$$w_i^t \leftarrow g_\theta^t(o_i^t, r^{t-1}) \quad (i = (1, 2, ..., n),\ r^0 = q) \tag{5}$$

$$\alpha_i^t \leftarrow \frac{\exp(\beta^t w_i^t)}{\sum_i \exp(\beta^t w_i^t)} \tag{6}$$

$$r^{t+1} \leftarrow \sum_i \alpha_i^t \cdot o_i^t \tag{7}$$

Updating component To forget the information already been used, we use updating component to renew the memory. It is replaced by the amount of unconsumed from the old one:

$$o_i^{t+1} \leftarrow (1 - \alpha_i^t)\, o_i^t \tag{8}$$

Contrary to other components, updating is not a mandatory component. When it is thought to have 1 hop, there is no need to use this.

Reasoning component Similar to attention component, reasoning component is also made up of MLP, represented as f_ϕ. It receives both q and the final result of attention component r_f and then take a softmax to produce the model answer $\hat{a}$:

$$\hat{a} \leftarrow \text{Softmax}(f_\phi(r^f, q)) \tag{9}$$

4 Experiments

4.1 Dataset

bAbI story-based QA dataset bAbI story-based QA dataset (Weston et al., 2015) is composed of 20 different type of tasks for testing natural language reasoning ability. Each task requires different methods to infer the answer. The dataset includes a set of statements, question and answer. A statement can be as short as two sentences and as long as 320 sentences. To answer the question, it is

necessary to find relevant statements to a given question and derive answer word from them. Answer is typically a single word but in a few tasks, answers are a set of words. Each task is regarded as success when the accuracy is greater than 95%. There are two versions of this dataset, one that has 1k training examples and the other with 10k examples.

bAbI dialog dataset bAbI dialog dataset (Bordes & Weston, 2016) is a set of 5 tasks within the goal-oriented context of restaurant reservation. It is designed to test if model can learn various abilities such as performing dialog management, querying knowledge bases (KBs), and interpreting the output of such queries. The KB can be queried using API calls and 4 fields (a type of cuisine, a location, a price range, and a party size) should be filled to issue an API call. Task 1 tests the capacity of interpreting a request and asking the right questions to issue an API call. Task 2 checks the ability to modify an API call. Task 3 and 4 tests the capacity of using outputs from an API call to propose options in the order of rating and to provide extra-information of what user asks for. Task 5 combines everything. As restaurant name, locations, and cuisine types always face to new entities, there are plain and OOV test sets to assess model's generalization ability. Training sets are 1k examples, which is small, to create realistic learning conditions.

4.2 Training details

bAbI story-based QA dataset We trained 2 hop RMN jointly on all tasks using 10k dataset for model to infer the solution suited to each type of tasks. We limited the input to the last 70 stories for all tasks except task 3 for which we limited input to the last 130 stories, similar to Sukhbaatar et al. (2015), Liu & Perez (2017), and Xiong et al. (2016). Then, we labeled each sentence with its relative position. Embedding component is similar to Santoro et al. (2017), where story and question are embedded through different LSTMs; 32 unit word-lookup embeddings; 32 unit LSTM for story and question. For attention component, as we use 2 hop RMN, there are g_θ^1 and g_θ^2; both are three-layer MLP consisting of 256, 128, 1 unit with ReLU activation function (Nair & Hinton, 2010). f_ϕ is composed of 512, 512, and 159 units (the number of words appeared in bAbI dataset is 159) of three-layer MLP with ReLU non-linearities where the final layer was a linear that produced logits for a softmax over the answer vocabulary. For regularization, we use batch normalization (Ioffe & Szegedy, 2015) for all MLPs. The softmax output was optimized with a cross-entropy loss function using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $2e^{-4}$.

bAbI dialog dataset We trained on full dialog scripts with every model response as answer, all previous dialog history as stories, and the last user utterance as question. Model selects the most probable response from 4,212 candidates which are ranked from a set of all bot utterances appearing in training, validation and test sets (plain and OOV) for all tasks combined. We also report results when we use match type features for dialog. Match type feature is an additional label on the candidates indicating if word is found on the dialog history. This feature can alleviate OOV problem. Training was done with Adam optimizer and a learning rate of $1e^{-4}$ for all tasks. Additional model details are given in Appendix A.

4.3 Results and Discussion

4.3.1 Time complexity

As attention component g_θ is applied to each object in memory referred in equation 5, the complexity reduces to $O(n)$. We compared processing time of a unit batch size 256 and the result is shown in figure 3. Because of the GPU memory exploding issue, there is no result for more than 170 sentences.

4.3.2 bAbI story-based QA

As we can see in table 1, RMN shows state-of-the-art result on bAbI story-based Question Answering dataset: 98.8% accuracy where only a single task with no catastrophic failure .

Table 2 shows how our model solved several tasks. RMN's attention component g_θ^1 and g_θ^2 complement each other to identify the necessary facts to answer correctly. Sometimes both g_θ^1 and g_θ^2 concentrate on the same stories which are all critical to answer the question, sometimes g_θ^1 finds a fact related to the given question and with this information g_θ^2 chooses the key fact to answer kitchen. While trained jointly, RMN learns these different solution for different task. For the task 3,

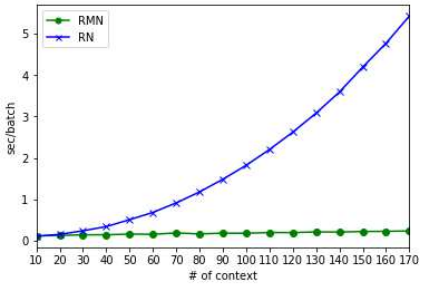


Figure 3: Time complexity of RN and RMN

Table 1: Results on bAbI story-based tasks with 10k training samples

Task	MemNN	MemN2N	GMemN2N	DMN	DMN+	DNC	EntNet	RN[1]	RMN
1: Single Supporting Fact	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
2: Two Supporting Facts	0.0	0.3	0.0	1.8	0.3	0.4	2.8	6.5	0.5
3: Three Supporting Facts	0.0	9.3	4.5	4.8	1.1	1.8	10.6	12.9	14.7
4: Two Argument Relations	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5: Three Argument Relations	2.0	0.6	0.2	0.7	0.5	0.8	0.4	0.5	0.4
6: Yes/No Questions	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0
7: Counting	15.0	3.7	1.8	3.1	2.4	0.6	0.8	0.2	0.5
8: Lists/Sets	9.0	0.8	0.3	3.5	0.0	0.3	0.1	0.1	0.3
9: Simple Negation	0.0	0.8	0.0	0.0	0.0	0.2	0.0	0.0	0.0
10: Indefinite Knowledge	2.0	2.4	0.2	2.5	0.0	0.2	0.0	0.0	0.0
11: Basic Coreference	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.4	0.5
12: Conjunction	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13: Compound Coreference	0.0	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0
14: Time Reasoning	1.0	0.0	0.0	0.0	0.0	0.4	3.6	0.0	0.0
15: Basic Deduction	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16: Basic Induction	0.0	0.4	0.0	0.6	45.3	33.1	52.1	50.3	0.9
17: Positional Reasoning	35	40.7	27.8	40.4	4.2	12.0	11.7	0.9	0.3
18: Size Reasoning	5.0	6.7	8.5	4.7	2.1	0.8	2.1	0.6	2.3
19: Path Finding	64.0	66.5	31.0	65.5	0.0	3.9	63.0	2.1	2.9
20: Agent's Motivations	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mean error (%)	6.7	6.6	3.7	6.4	2.8	2.7	7.4	3.7	**1.2**
Failed tasks (err. >5%)	4	4	3	2	**1**	2	4	3	**1**

the only failed task, attention component still functions well; focuses sequentially on the supporting sentences. However, the reasoning component, f_ϕ, had a difficulty on catching a word 'before'. We could easily figure out 'before' implies 'just before' the certain situation, whereas RMN confused its meaning. As shown in table 2c, our model found all previous locations before the garden. Still, it is remarkable that the simple MLP carried out all of these various roles.

4.3.3 bAbI DIALOG

Results are given in Table 3. RMN outperforms previous memory-augmented models with a significant difference without given any match type. Unlike previous models, match type features are helpful only on OOV tasks. This is mainly attributed to the impressive result on task 4. Table 4c shows how RMN solved task 4. The attention component adequately focused on necessary knowledge base and the reasoning component inferred appropriate answer even on the OOV task. It implies f_ϕ is able to leverage information coming from the entities whether they are seen or unseen in the training dialog.

On the other hand, f_ϕ failed to perform well on the OOV task 1 and 2 even though g_θ^1 properly focused on the related object. We state that this contrast is originated from the information contained in one sentence is relatively large in task 1 and 2. Consider the example in Table 4a and Table 4c. Supporting sentence of task 4 have one keyword out of three words, whereas supporting sentences of task 1 and 2 consist of four keywords (cuisine, location, number and price) out of sixteen words. This confuses RMN to recognize critical information to choose a right answer.

[1]Our implementation. The result is different from what Santoro et al. (2017) mentioned, which is caused by the initialization.

Table 2: bAbI story-based task visualization of α

(a) Task 7

Seq.	**Task 7: Counting**	α^1	α^2
8	John grabbed the apple there .	0.02	0.00
9	John gave the apple to Mary .	0.08	0.16
10	Mary passed the apple to John .	0.17	0.24
11	Mary journeyed to the hallway .	0.00	0.01
13	Sandra went to the garden .	0.00	0.00
14	Mary went to the kitchen .	0.00	0.03
15	Mary picked up the football there .	0.29	0.24
16	Mary picked up the milk there .	0.27	0.32
User input	How many objects is Mary carrying?		
Answer	Two		
Model answer	Two **[Correct]**		

(b) Task 14

Seq.	**Task 14: Time reasoning**	α^1	α^2
1	Mary went back to the school yesterday .	0.00	0.01
2	Fred went to the school yesterday .	0.00	0.00
3	Julie went back to the kitchen yesterday .	0.13	0.98
4	Fred journeyed to the kitchen this morning .	0.00	0.00
5	This morning julie journeyed to the school .	0.66	0.02
6	This evening mary went back to the school .	0.01	0.00
7	This afternoon julie went to the bedroom .	0.06	0.00
User input	Where was Julie before the school ?		
Answer	Kitchen		
Model answer	Kitchen **[Correct]**		

(c) Task 3

Seq.	**Task 3: Three Supporting Facts**	α^1	α^2
33	Daniel took the football .	0.46	0.01
39	Sandra travelled to the bedroom .	0.01	0.00
40	Daniel moved to the bathroom .	0.01	0.31
41	Sandra got the milk .	0.00	0.00
42	Daniel travelled to the garden .	0.04	0.42
46	Daniel went to the hallway .	0.00	0.25
50	Daniel put down the apple .	0.00	0.01
51	Daniel put down the football there .	0.38	0.00
User input	Where was the football before the garden ?		
Answer	Bathroom		
Model answer	Bathroom **[Correct]**		

Seq.	**Task 3: Three Supporting Facts**	α^1	α^2
1	Mary got the football .	0.26	0.00
3	Mary picked up the football .	0.39	0.00
5	Mary moved to the office .	0.00	0.05
6	Mary went to the hallway .	0.00	0.05
11	Mary travelled to the garden .	0.00	0.21
12	Mary travelled to the kitchen .	0.00	0.29
14	Mary moved to the office .	0.00	0.25
16	Mary went to the garden .	0.00	0.05
22	Mary discarded the football there .	0.24	0.00
User input	Where was the football before the garden ?		
Answer	Office		
Model answer	Kitchen **[Incorrect]**		

Table 3: Results on bAbI dialog tasks [2]

Task	Plain			With Match		
	MemN2N	GMemN2N	RMN	MemN2N	GMemN2N	RMN
1: Issuing API calls	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
2: Updating API calls	**0.0**	**0.0**	**0.0**	1.7	**0.0**	**0.0**
3: Displaying options	**25.1**	**25.1**	**25.1**	**25.1**	**25.1**	**25.1**
4: Providing extra information	40.5	42.8	**0.0**	**0.0**	**0.0**	**0.0**
5: Conducting full dialogs	3.9	3.7	**2.8**	6.6	**2.0**	2.8
Average error rates (%)	13.9	14.3	**5.6**	6.7	**5.4**	5.6
1 (OOV): Issuing API calls	27.7	17.6	**16.8**	3.5	**0.0**	**0.0**
2 (OOV): Updatining API calls	**21.1**	**21.1**	**21.1**	5.5	5.8	**0.0**
3 (OOV): Displaying options	25.6	**24.7**	24.9	**24.8**	24.9	25.1
4 (OOV): Providing extra information	42.4	43.0	**0.0**	**0.0**	**0.0**	**0.0**
5 (OOV): Conducting full dialogs	34.5	**33.3**	34.6	22.3	**20.6**	21.2
Average error rates (%)	30.3	27.9	**19.5**	11.2	10.3	**9.3**

The main goal of task 3 is to recommend restaurant from knowledge base in the order of rating. Different from other tasks, RMN yields same error rate 25.1% with MemN2N and GMemN2N. All failed cases are displaying restaurant where the user input is <silence>which is somewhat ambiguous trigger to find question relevant information. As shown in Table 4b, there are two different types of response to the same user input. One is to check whether all the required fields are given from the previous dialog and then ask user for the missing fields or send a "Ok let me look into some options for you." message. The other type is to recommend restaurant starts from the highest rating. Three models show lack of ability to discriminate these two types so that concluded to the same results. To verify our statement, we performed additional experiment on task 3 and check the performance gain (extra result is given in Table 6 of Appendix B).

5 Conclusion

Our work, RMN, is a simple and powerful architecture for relational reasoning that restructured RN module as 4 components of Memory Network. We found the similarity between the role of output feature map of Memory Network to read relevant object given the input object and g_θ of RN to infer if two objects are related. From this interesting discovery, we modified g_θ as an attention component and raised the state-of-the-art performance on story-based QA and goal-oriented dialog dataset. It is

[2] we only compare with models under the same conditions

remarkable that RMN solved dialog task 4 in 100% accuracy with and without match, and even on the oov test dataset.

A key contribution of RMN is it reduces RN's problematic quadratic complexity to linear complexity which enables applications on broader domains while preserving RN's simplicity .

Future work will apply RMN to image based reasoning task (e.g., CLEVR). To extract features from the image, VGG net (Simonyan & Zisserman, 2014) is used in convention and outputs 196 objects of 512 dimensional vector which also requires large sized memory. An important direction will be to find an appropriate way to focus sequentially on related image object which was rather easy in text based reasoning.

Table 4: Visualization of α on bAbI dialog task 1, 3, and 4 without match type

(a) Plain and OOV result of task 1

Seq.	Locutor	**Task 1: Issuing API calls**	α^1
1	user	Good morning	0.01
2	bot	Hello what can i help you with today	0.01
3	user	Can you make restaurant reservation for eight people in cheap price range with british cuisine	0.23
4	bot	I'm on it	0.00
5	user	<silence>	0.14
6	bot	Where should it be	0.01
7	user	In madrid	0.25
8	bot	Ok let me look into some options for you	0.18
User input		<silence>	
Answer		API_call british madrid eight cheap	
Model answer		API_call british madrid eight cheap **[Correct]**	

Seq.	Locutor	**Task 1 (OOV): Issuing API calls**	α^1
1	user	Hi	0.01
2	bot	Hello what can i help you with today	0.01
3	user	May I have table with cantonese cuisine in cheap price range in seoul for two people	0.49
4	bot	I'm on it	0.10
5	user	<silence>	0.01
6	bot	Ok let me look into some options for you	0.27
User input		<silence>	
Answer		API_call cantonese seoul two cheap	
Model answer		API_call italian paris two cheap **[Incorrect]**	

(b) Same user input with different goal in task 3

Seq.	Locutor	**Task 3: Displaying options**	α^1
7	user	resto_1 r_rating 1	0.01
14	user	resto_8 r_rating 8	0.01
21	user	resto_3 r_rating 3	0.01
22	user	Hello	0.01
23	bot	Hello what can i help you with today	0.01
24	user	Can you book table with french food for two people in madrid in moderate price range	0.21
25	bot	I'm on it	0.01
User input		<silence>	
Answer		Where should it be?	
Model answer		Where should it be? **[Correct]**	

Seq.	Locutor	**Task 3: Displaying options**	α^1
7	user	resto_1 r_rating 1	0.00
14	user	resto_8 r_rating 8	0.00
21	user	resto_3 r_rating 3	0.00
24	user	Can you book table with french food for two people in madrid in moderate price range	0.14
25	bot	I'm on it	0.00
26	user	<silence>	0.09
27	bot	Ok let me look into some options for you	0.15
28	user	<silence>	0.10
29	bot	What do you think of this option : resto_8	0.08
30	user	No this does not work for me	0.06
31	bot	Sure let me find other option for you	0.08
User input		<silence>	
Answer		What do you think of this option: resto_3	
Model answer		What do you think of this option: resto_7 **[Incorrect]**	

(c) Plain and OOV result of task 4

Seq.	Locutor	**Task 4: Providing extra information**	α^1
1	user	resto_3 r_phone resto_3_phone	0.77
3	user	resto_3 r_address resto_3_address	0.01
4	user	resto_3 r_location london	0.01
6	user	resto_3 r_price cheap	0.01
10	user	I'd like to book table at resto_3	0.02
11	bot	Great let me do reservation	0.03
User input		What is phone number of restaurant?	
Answer		Here it is: resto_3_phone	
Model answer		Here it is: resto_3_phone **[Correct]**	

Seq.	Locutor	**Task 4 (OOV): Providing extra information**	α^1
1	user	resto_1 r_phone resto_1_phone	0.79
3	user	resto_1 r_address resto_1_address	0.01
4	user	resto_1 r_location hanoi	0.01
6	user	resto_1 r_price expensive	0.01
10	user	Can you make restaurant reservation at resto_1	0.01
11	bot	Great let me do reservation	0.04
User input		What is phone number of restaurant?	
Answer		Here it is: resto_3_phone	
Model answer		Here it is: resto_3_phone **[Correct]**	

REFERENCES

Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683, 2016.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

S. Chopra J. Weston and A. Bordes. Memory networks. *International Conference on Learning Representations (ICLR)*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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 *International Conference on Machine Learning*, pp. 1378–1387, 2016.

Fei Liu and Julien Perez. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pp. 1–10, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*, pp. 2397–2406, 2016.

A MODEL DETAILS

Table 5: Hyperparameters of Related Memory Networks on bAbI dialog tasks

Task	Story and Question Embedding	Word-lookup Embedding Dim	Hop	g_θ	f_ϕ	Activation	Use Batch Norm
1	sum	128	1	2048, 2048, 1	2048, 2048, 4212	tanh	True
2	sum	128	1	1024, 1024, 1	1024, 1024, 4212	tanh	True
3	sum	128	1	1024, 1024, 1024, 1	1024, 1024, 1024, 4212	tanh	True
4	concatenation	50	1	1024, 1024, 1	1024, 1024, 4212	tanh	True
5	concatenation	64	1	4096, 4096, 1	4096, 4096, 4212	tanh	True

B ADDITIONAL RESULTS

Table 6: Visualization of α^1 and α^2 on user input revised bAbI dialog task 3 without match type

Seq.	Locutor	Task 3: Displaying options	α^1	α^2
1	user	resto_8 r_phone resto_8_phone	0.00	0.01
2	user	resto_8 r_cuisine french	0.01	0.01
3	user	resto_8 r_address resto_8_address	0.01	0.01
4	user	resto_8 r_location madrid	0.01	0.01
5	user	resto_8 r_number two	0.00	0.01
6	user	resto_8 r_price moderate	0.00	0.01
7	user	resto_8 r_rating 8	0.01	0.39
8	user	resto_3 r_phone resto_3_phone	0.01	0.01
9	user	resto_3 r_cuisine french	0.00	0.01
10	user	resto_3 r_address resto_3_address	0.00	0.01
11	user	resto_3 r_location madrid	0.00	0.01
12	user	resto_3 r_number two	0.00	0.01
13	user	resto_3 r_price moderate	0.00	0.01
14	user	resto_3 r_rating 3	0.00	0.01
15	user	resto_1 r_phone resto_1_phone	0.01	0.01
16	user	resto_1 r_cuisine french	0.01	0.01
17	user	resto_1 r_address resto_1_address	0.01	0.01
18	user	resto_1 r_location madrid	0.01	0.01
19	user	resto_1 r_number two	0.00	0.01
20	user	resto_1 r_price moderate	0.00	0.01
21	user	resto_1 r_rating 1	0.00	0.01
22	user	Good morning	0.01	0.01
23	bot	Hello what can I help you with today?	0.12	0.01
24	user	May I have table for two with french food?	0.08	0.02
25	bot	I'm on it	0.03	0.01
26	user	<silence>	0.01	0.01
27	bot	Where should it be	0.09	0.01
28	user	In madrid	0.01	0.01
29	bot	Which price range are looking for?	0.09	0.01
30	user	In moderate price range please	0.01	0.01
31	bot	Ok let me look into some options for you	0.24	0.01
User input		<silence><silence>		
Answer		What do you think of this option: resto_8		
Model answer		What do you think of this option: resto_8 **[correct]**		

We modify the user input from <silence>to <silence><silence>when looking for restaurant recommendations. This makes model to distinguish two different situations whether to ask for additional fields or to recommend restaurant.