Dialogue over Context and Structured Knowledge using a Neural Network Model with External Memories

Yuri Murayama **Lis Kanashiro Pereira** **Ichiro Kobayashi**
Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo, Japan
{murayama.yuri, koba}@is.ocha.ac.jp
kanashiro.pereira@ocha.ac.jp

Abstract

The Differentiable Neural Computer (DNC), a neural network model with an addressable external memory, can solve algorithmic and question answering tasks. There are various improved versions of DNC, such as rsDNC and DNC-DMS. However, how to integrate structured knowledge into these DNC models remains a challenging research question. We incorporate an architecture for knowledge into such DNC models, i.e. DNC, rsDNC and DNC-DMS, to improve the ability to generate correct responses using both contextual information and structured knowledge. Our improved rsDNC model improves the mean accuracy by approximately 20% to the original rsDNC on tasks requiring knowledge in the dialog bAbI tasks. In addition, our improved rsDNC and DNC-DMS models also yield better performance than their original models in the Movie Dialog dataset.

1 Introduction

Recently, deep neural networks have made significant progress in complex pattern matching of various tasks such as computer vision and natural language processing. However, these models are limited in their ability to represent data structures such as graphs and trees, to use variables, and to handle representations over long sequences. Recurrent neural networks (RNNs) can capture the long-range dependencies of sequential data and are also known as Turing-Complete (Siegelmann and Sontag, 1995) and therefore are capable of simulating arbitrary procedures, if properly wired. However, RNNs struggled with the vanishing gradients problem (Bengio et al., 1994). The long short-term memory (LSTM) architecture addressed this problem by taking gating mechanisms into RNN architectures and calculating the gradients by element-wise multiplication with the gate value at every time-step (Hochreiter and Schmidhuber, 1997). LSTMs became quite successful and helped to outperform traditional models because of the sequence to sequence model (Sutskever et al., 2014) and attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015). Yet LSTM based models have not reached a real solution to the problems mentioned as the limitations of deep neural networks.

On the other hand, in the von Neumann architecture, programs can be run by three fundamental mechanisms: a processing unit that performs simple arithmetic operations and logic ones, a control unit that takes control flow instructions such as sequential execution, conditional branch and loop, and a memory unit that data and instructions are written to and read from during computation (Neumann, 1945). This architecture can represent complex data structures and learn to perform algorithmic tasks. It separates computation by a processor (i.e. a processing unit and a control one) and memory, whereas neural networks mix computation and memory into the network weights. To improve the performance of standard neural networks, Graves et al. (2014) proposed a neural network model with an addressable external memory called Neural Turing Machine (NTM). The whole architecture of NTM is differentiable, therefore can be trained end-to-end including how to access to the memory. Further, Graves et al. (2016) improved the memory access mechanism and proposed Differentiable Neural Computer (DNC). It solved algorithmic tasks over structured data such as traversal and shortest-path tasks of a route map and an inference task of a family tree. In the experiment on question answering with premises, input sequences were written to the memory and necessary information to infer the answer was read from the memory, hence representing variables. DNC was also able to learn long sequences by the dynamic memory access mechanism. There are various improved versions of DNC, such as rsDNC (Franke et al., 2018) and DNC-DMS (Csordás and Schmidhuber, 2019).

However, how to integrate structured knowledge into these DNC models remains a challenging research question. This paper investigates how to incorporate structured knowledge into DNC models. We extend single-memory unit DNC models to a multiple-memory architecture that leverages both contextual and structured knowledge information. We add an extra memory unit that encodes knowledge from knowledge bases. In contrast with RNNs, the memory-augmentation of DNC allows an explicit storage and manipulation of complex data structures over a long time-scale (Franke et al., 2018). Our main contributions are as follows:

- We incorporate a knowledge memory architecture into DNC models, i.e. DNC, rsDNC and DNC-DMS, to improve the ability to generate correct responses using both contextual information and structured knowledge.

- Our improved rsDNC model improves the mean accuracy by approximately 20% to the original rsDNC on tasks requiring knowledge in the dialog bAbI tasks (Bordes and Weston, 2016).

- In addition, our improved rsDNC and DNC-DMS models also yield better performance than their original models in the Movie Dialog dataset (Dodge et al., 2016).

The whole paper is organized as follows. Section 2 briefly introduces the DNC, rsDNC and DNC-DMS models. We describe our proposed model in Section 3 and our experiments and detailed analysis in Section 4. Section 5 introduces related works. Finally, we conclude this paper and explore the future work in Section 6.

2 Differentiable Neural Computer

The Differentiable Neural Computer (DNC) is a neural network coupled to an external memory matrix $M \in \mathbb{R}^{N \times W}$, as shown in Figure 1. It uses attention mechanisms to define weightings over N locations of the memory matrix M that represent which locations should be read or written mainly.

For the read operation, the read vector $\boldsymbol{r}$ is computed as a weighted sum over the memory locations

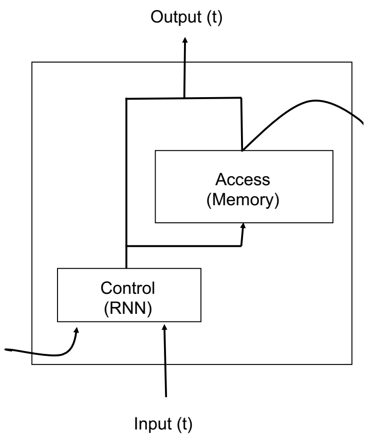

Figure 1: Overview of DNC

by applying a read weighting $\boldsymbol{w}^r$ over memory M:

$$\boldsymbol{r} = \sum_{i=1}^{N} M[i, \cdot]\boldsymbol{w}^r[i]$$

where the '$\cdot$' denotes all $j = 1, ..., W$.

For the write operation, the memory M is modified by using a write weighting $\boldsymbol{w}^w$ to first erase with an erase vector $\boldsymbol{e}$, then add a write vector $\boldsymbol{v}$:

$$M[i, j] \leftarrow M[i, j](1 - \boldsymbol{w}^w[i]\boldsymbol{e}[j]) + \boldsymbol{w}^w[i]\boldsymbol{v}[j]$$

The weightings are defined by the following three attention mechanisms:

- Content-based addressing: compares a key vector to the content of each location in memory and calculates similarity scores to define a read weighting for associative recall or a write weighting to modify a relevant vector in memory.

- Temporal memory linkage: records tracks of consecutively written memory locations to be able to read sequences in the order of which locations were written to.

- Dynamic memory allocation: frees and allocates memory as needed for writing by representing the degree of each location usage which can be increased with each write and decreased after each read and reallocating memory with a low degree of usage.

The whole system is differentiable and can be learned with gradient descent.

Recently, variations of the DNC model have been proposed, such as the robust and scalable DNC (rsDNC) (Franke et al., 2018) and the DNC-DMS (Csordás and Schmidhuber, 2019).

Robust and Scalable DNC : Focusing on QA tasks, Franke et al. (2018) extended the DNC to be more robust and scalable (rsDNC), with the following four improvements: (1) using only the content-based memory unit to reduce memory consumption and training time, (2) applying layer normalization to lower the variance in performance between different runs, (3) using bypass dropout to make the memory unit's effect stronger, and (4) introducing a bidirectional architecture to encode input sequences in a more informative way.

DNC-DMS : Csordás and Schmidhuber (2019) tackled three problems of vanilla DNC and proposed an improved model called DNC-DMS. First, the lack of key-value separation makes the content-based address distribution flat and prevents the model from accessing specific parts of a memory. By masking improper parts of both look-up key and memory content, the key-value separation can be controlled dynamically. Second, memory de-allocation mechanisms do not affect memory content which is crucial to content-based addressing and result in memory aliasing. Thus, DNC-DMS proposed to erase memory content completely. Lastly, chaining multiple reads with the temporal linkage matrix exponentially blurs the address distribution. Exponentiation and re-normalization of the distribution reduced the effect of exponential blurring and improved sharpness of the link distribution.

Although these methods lead to good improvements over the original DNC model, none of them addressed how to incorporate structured knowledge into the DNC explicitly.

3 Proposed Method

We expand three models, DNC, rsDNC, and DNC-DMS, by adding an extra memory architecture to store structured knowledge. Therefore, our proposed model consists of a control unit and two memory units. One memory unit stores contextual information in the dialogue and we call it "context memory". The other memory unit stores knowledge information and we call it "knowledge memory". The differences from the original DNC models are to introduce the knowledge memory and add the operation for it. Figure 2 shows the overview of our proposed model based on DNC.

The procedures at every time-step t are described as follows:

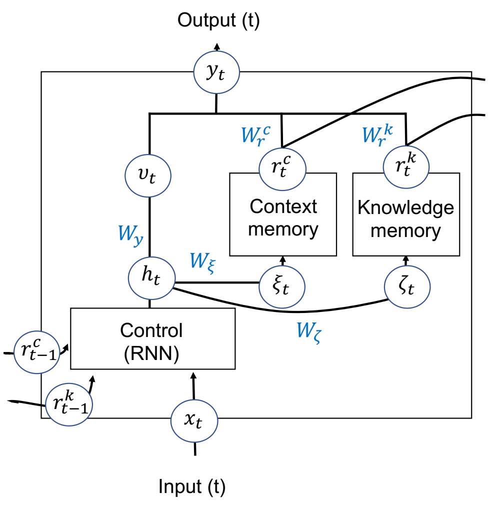

Figure 2: Overview of our proposed model based on DNC

1. The controller (RNN) receives an input vector $\boldsymbol{x}_t$, a set of R read vectors $\boldsymbol{r}_{t-1}^c = [\boldsymbol{r}_{t-1}^{c,1}; ...; \boldsymbol{r}_{t-1}^{c,R}]$ ($\boldsymbol{r}_{t-1}^c$ is a concatenation of $\boldsymbol{r}_{t-1}^{c,1}, ..., \boldsymbol{r}_{t-1}^{c,R}$) from the context memory matrix $M_{t-1}^c \in \mathbb{R}^{N \times W}$ at the previous time-step and a set of R read vectors $\boldsymbol{r}_{t-1}^k = [\boldsymbol{r}_{t-1}^{k,1}; ...; \boldsymbol{r}_{t-1}^{k,R}]$ from the knowledge memory matrix $M_{t-1}^k \in \mathbb{R}^{N' \times W'}$ at the previous time-step. It then emits an hidden vector $\boldsymbol{h}_t$.

2. By linear transformation of $\boldsymbol{h}_t$, an output vector $\boldsymbol{v}_t = W_y \boldsymbol{h}_t$, an interface vector $\boldsymbol{\xi}_t = W_\xi \boldsymbol{h}_t$ that parameterizes the context memory interactions at the current time-step and an interface vector $\boldsymbol{\zeta}_t = W_\zeta \boldsymbol{h}_t$ for the knowledge memory are obtained. The W terms denote learnable weight matrices.

3. The write operation to the context memory is performed using $\boldsymbol{\xi}_t$ and its state is updated. The write operation to the knowledge memory is not performed.

4. Finally, the output vector $\boldsymbol{y}_t$ is calculated by adding $\boldsymbol{v}_t$ to a vector obtained by multiplying the concatenation of the current read vectors $\boldsymbol{r}_t^c$ from the context memory and W_r^c and a vector obtained by multiplying the concatenation of the current read vectors $\boldsymbol{r}_t^k$ from the knowledge memory and W_r^k.

$$\boldsymbol{y}_t = \boldsymbol{v}_t + W_r^c \boldsymbol{r}_t^c + W_r^k \boldsymbol{r}_t^k$$

The read vectors $\boldsymbol{r}_t^c$ and $\boldsymbol{r}_t^k$ are appended to the controller input at the next time-step.

The read and write operations to two memories are performed by repeating the above procedures.

To build the knowledge memory unit used in our models, we first run the original DNC model on a knowledge base (KB). We then use the pre-trained memory unit as the knowledge memory unit in our proposed models. The process to build this knowledge memory is described in the next section.

Knowledge Memory Building We built a knowledge memory unit with a knowledge base (KB). Facts in the KB have a Resource Description Framework (RDF)[1] triple structure "(subject, relation, object)". For example, information such as "Washington, D.C. is the capital of the U.S." is expressed as `(the U.S., capital, Washington, D.C.)`.

We applied the original DNC model, which has a single memory, to learn KB facts by giving the model all three components of a triple and any two of a triple and then making the model learn to return the other of a triple. For instance, when inputs for the model are `` the U.S.'', `` capital'', `` Washington, D.C.'', `` the U.S.'', and ``capital'', the output is `` Washington, D.C.''. The model was trained using all triples of the KB and produced a memory unit which stores the whole KB. We used this pre-trained memory unit as the knowledge memory unit in our proposed models.

We trained the DNC model using memory dimensions of 512×128 because the results of our proposed models were better than when we also trained using memory dimensions of 256×64 which were the same as context memory dimensions in our proposed method. Whereas the context memory just stores each dialogue content in the dataset, the knowledge memory stores the whole content of the KB and thus it is reasonable that the knowledge memory needs to be larger than the context memory. We evaluated the model with the accuracy and used TransE (Bordes et al., 2013) for KB's word embeddings.

4 Experiments

We evaluated our approach on two dialogue datasets, the (6) dialog bAbI tasks (Bordes and Weston, 2016) and the Movie Dialog dataset (Dodge et al., 2016). Both datasets require context comprehension and knowledge background, and provide dialogue data on a specific domain and RDF triple data to answer questions in the dialogue.

[1]https://www.w3.org/TR/rdf11-primer/

4.1 Implementation Details

The hyperparameters of all models are mainly based on the original DNC paper (Graves et al., 2016). We trained all models using one layer LSTM (Hochreiter and Schmidhuber, 1997) with a hidden layer of size 256 , a batch of size 32, a learning rate of 1×10^{-4}, context memory dimensions of 256×64, knowledge memory dimensions of 512×128, four read heads, one write head. We used the RMSProp optimizer (Tieleman and Hinton, 2012) with a momentum of 0.9. rsDNC models have a dropout probability of 10%, following (Franke et al., 2018). We used TransE (Bordes et al., 2013) for KB's word embeddings and GloVe embeddings (Pennington et al., 2014) for words that do not appear in the KB but appear in the dialogue such as "the" and "what" referring to (Saha et al., 2018). The dimension of each word embedding vector is 200. We stopped training if the result of a validation set drops ten epochs in a row and the model repeats this five times during training. We run every model three times under different random initializations and report the averaged results.

4.2 Dialog bAbI tasks

The (6) dialog bAbI tasks (Bordes and Weston, 2016) are a set of six dialogue tasks within the goal-oriented context of restaurant reservation. Among them, we focus on Task 5, which combines Tasks 1-4 to generate full dialogs. We also removed sentences starting with the special token "api_call" from it in our work. Training, validation and test sets hold 1,000 examples, respectively. It also includes an Out-Of-Vocabulary (OOV) test set of 1,000 examples that include entities unseen in training and validation sets. The KB contains 8,400 facts in the restaurant domain such as ``resto_seoul_cheap_korean_1stars, R_cuisine, korean''. The number of entities is 3,635, the number of relations is 7, and the vocabulary size of the dialog is 2,043.

Results In Table 1, we present the mean per-response accuracy over three different runs for all models. For clarity, we use the following notation: DNC is the original DNC model, rsDNC refers to the robust and scalable DNC model proposed by Franke et al. (2018), DNC-DMS (Csordás

Task	DNC	DNC+KM	rsDNC	rsDNC+KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
Full dialogs	83.04%	**86.52%**	87.77%	**92.46%**	**84.48%**	83.83%	82.50%	**84.30%**
Full dialogs (OOV)	73.06%	**74.60%**	**76.11%**	75.90%	**73.57%**	71.83%	72.33%	**72.52%**

Table 1: Mean per-response accuracy of different models and our proposed models in the dialog bAbI tasks (Task 5).

Task	DNC	DNC+KM	rsDNC	rsDNC+KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
w/o KB facts	99.99%	99.99%	100.00%	100.00%	100.00%	100.00%	99.99%	**100.00%**
w/ KB facts	29.53%	**43.98%**	49.14%	**68.66%**	**35.49%**	32.78%	27.28%	**34.73%**
w/o KB facts (OOV)	95.98%	**98.01%**	**99.99%**	99.72%	**96.65%**	94.37%	95.03%	**95.27%**

Table 2: Detailed results in the dialog bAbI tasks (Task 5). w/o KB facts denotes tasks that can be answered without KB facts and w/ KB facts denotes tasks that need KB facts to answer questions. The results of w/ KB facts (OOV) are omitted since they are all 0.00%.

and Schmidhuber, 2019) has three modifications (i.e. de-allocation mechanisms, masked content based addressing, and sharpness enhancement), and DNC-MD (a variant of the DNC-DMS model) is with only masking and de-allocation modifications. Models with "+KM" notation are our proposed DNC models with the added knowledge memory unit.

DNC+KM outperformed the original DNC on both Full dialogs and OOV tasks. Table 2 shows the results in detail separating tasks that can be answered without KB facts and tasks that require KB facts to answer questions. The w/o KB facts task has 12,351 sentences and the w/ KB facts task contains 3,912 sentences in the test set. Though DNC and DNC+KM were the same on the w/o KB facts task (99.9%), there was a significant improvement of 14.45% on the w/ KB facts task, where the DNC and DNC+KM models obtained accuracy scores of 29.53% and 43.98%, respectively.

rsDNC+KM achieved the best performance overall and also improved the results on the w/ KB facts task by 19.52%, where the rsDNC and rsDNC+KM models obtained accuracy scores of 68.66% and 49.14%, respectively. Focusing on the rsDNC+KM model, we visualized the results of read/write attention weights from/to memories at every time-step to investigate what the model wrote to memories and what it read from memories. Figure 3, Figure 4, and Figure 5 shows the attention weights visualization on a successful example where the outputs of the model are all correct, as presented in Table 3. In Figure 3, the horizontal axis represents locations in the context memory and the vertical axis represents inputs of data, or user's utterances and outputs of the model at every time-step. While the model takes input sequences, it returns nothing, and while there is no input, it generates responses, in other words, when input sequences are "can, you, make, a, restaurant, reservation, in, paris, -, -, -", the output sequences are "-, -, -, -, -, -, -, -, i'm, on, it" ('-' is a padding word). Figure 3 shows write attention weights to the context memory corresponding to turn 2 to turn 6 in the example shown in Table 3. There are strong attentions on KB words such as "paris" and it is interesting that there are also attentions on words used to change the conditions of a restaurant such as "actually" and "instead". Figure 4 shows read attention weights from the context memory corresponding to turn 8 to 14 in the example shown in Table 3. The model's memory uses four read heads and we show one of them. The slot where "indian" was strongly written to has an attention when the model outputs restaurant information. The correct answer is an indian restaurant and therefore read information of "indian" is thought to be useful. Figure 5 presents read attention weights from the knowledge memory. There are attentions before the model answers restaurant information and distinctive features are not found compared to the context memory.

We also examined a poor performance example where the model made mistakes in all w/ KB facts tasks, as shown in Table 4. Figure 6 shows a part of the results of write attention weights to the context memory between turn 2 and 8 in Table 4. KB words such as "moderate" and "british" have strong attentions. Figure 7 presents a part of read attention weights from the context memory between turn 10 and 14 in Table 4. There are attentions on a slot where "british" was written to and on another slot where "with" was written to when

Turn	User's utterance	Outputs of rsDNC+KM	Target
1	good morning	hello what can i help you with today ✓	hello what can i help you with today
2	can you make a restaurant reservation in paris for six with indian cuisine	i'm on it ✓	i'm on it
3	<silence>	which price range are looking for ✓	which price range are looking for
4	i am looking for a expensive restaurant	ok let me look into some options for you ✓	ok let me look into some options for you
5	actually i would prefer in madrid	sure is there anything else to update ✓	sure is there anything else to update
6	instead could it be in a moderate price range	sure is there anything else to update ✓	sure is there anything else to update
7	no	ok let me look into some options for you ✓	ok let me look into some options for you
8	<silence>	what do you think of this option: resto_madrid_moderate_indian_7stars ✓	what do you think of this option: resto_madrid_moderate_indian_7stars
9	no i don't like that	sure let me find an other option for you ✓	sure let me find an other option for you
10	<silence>	what do you think of this option: resto_madrid_moderate_indian_5stars✓	what do you think of this option: resto_madrid_moderate_indian_5stars
11	no i don't like that	sure let me find an other option for you ✓	sure let me find an other option for you
12	<silence>	what do you think of this option: resto_madrid_moderate_indian_2stars ✓	what do you think of this option: resto_madrid_moderate_indian_2stars
13	let's do it	great let me do the reservation ✓	great let me do the reservation
14	may i have the phone number of the restaurant	here it is resto_madrid_moderate_indian_2stars_phone ✓	here it is resto_madrid_moderate_indian_2stars_phone
15	you rock	is there anything i can help you with✓	is there anything i can help you with
16	no thanks	you're welcome ✓	you're welcome

Table 3: Outputs of rsDNC+KM on a successful example on the dialog bAbI tasks

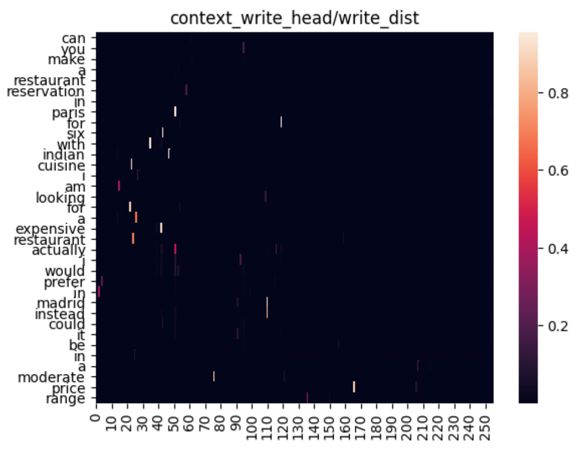

Figure 3: Visualized result of write attention weights to the context memory in a rsDNC+KM's successful example on the dialog bAbI tasks. The horizontal axis represents locations in the context memory and the vertical axis represents inputs of data and outputs of the model at every time-step.

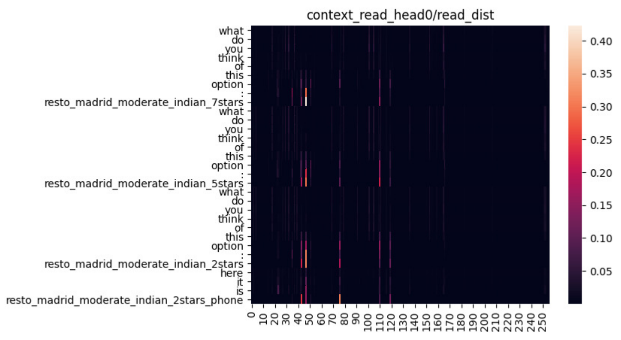

Figure 4: Visualized result of read attention weights from the context memory in the rsDNC+KM's successful example on the dialog bAbI tasks

the model outputs restaurant information. Unnecessary information of "with" may be a negative influence on the outputs of the model. Figure 8 shows read attention weights from the knowledge memory. There are attentions before the model answers restaurant information and it shows a similar behavior to read attention weights from the knowledge memory in the good example. Considering the visualized results, the contributions of the knowledge memory appear blurry (it might give the model some information about when to answer KB entities), however, the performance of rsDNC improves by adding the knowledge memory architecture and therefore it is our future work to investigate the knowledge memory behavior in more detail.

Regarding DNC-DMS models, the scores of DNC-DMS+KM were lower than the ones of DNC-DMS. We also conducted experiments on DNC-MD models since Csordás and Schmidhuber (2019) reported that DNC-MD performed better than DNC-DMS on QA tasks, and we found that the performance of the DNC-MD+KM model was higher than DNC-MD. We hypothesize this is due to the nature of the knowledge memory, which does not use the temporal memory linkage. The order of the KB triples when building knowledge memory and the order of the words in the dialogue are irrelevant, therefore sharpness enhancement does not work well.

Turn	User's utterance	Outputs of rsDNC+KM	Target
1	good morning	hello what can i help you with today ✓	hello what can i help you with today
2	i'd like to book a table in a moderate price range	i'm on it ✓	i'm on it
3	<silence>	any preference on a type of cuisine ✓	any preference on a type of cuisine
4	with british food	where should it be ✓	where should it be
5	madrid please	how many people would be in your party ✓	how many people would be in your party
6	for four people please	ok let me look into some options for you ✓	ok let me look into some options for you
7	instead could it be in a expensive price range	sure is there anything else to update ✓	sure is there anything else to update
8	actually i would prefer for six	sure is there anything else to update ✓	sure is there anything else to update
9	no	ok let me look into some options for you ✓	ok let me look into some options for you
10	<silence>	what do you think of this option: resto_london_expensive_british_4stars ✗	what do you think of this option: resto_madrid_expensive_british_8stars
11	no this does not work for me	sure let me find an other option for you ✓	sure let me find an other option for you
12	<silence>	what do you think of this option: resto_paris_expensive_british_4stars ✗	what do you think of this option: resto_madrid_expensive_british_6stars
13	i love that	great let me do the reservation ✓	great let me do the reservation
14	do you have its phone number	here it is resto_madrid_moderate_british_5stars_phone ✗	here it is resto_madrid_expensive_british_6stars_phone
15	thanks	is there anything i can help you with✓	is there anything i can help you with
16	no thank you	you're welcome ✓	you're welcome

Table 4: Outputs of rsDNC+KM on a poor performance example on the dialog bAbI tasks

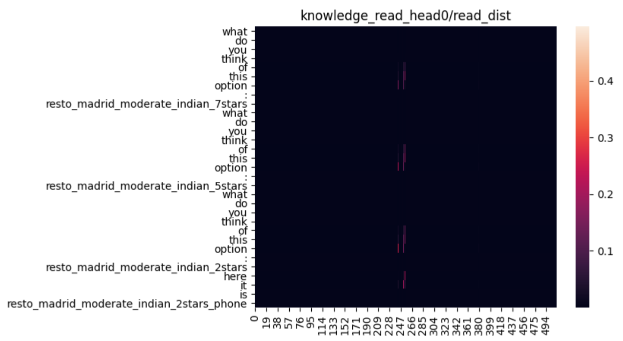

Figure 5: Visualized result of read attention weights from the knowledge memory in the rsDNC+KM's successful example on the dialog bAbI tasks

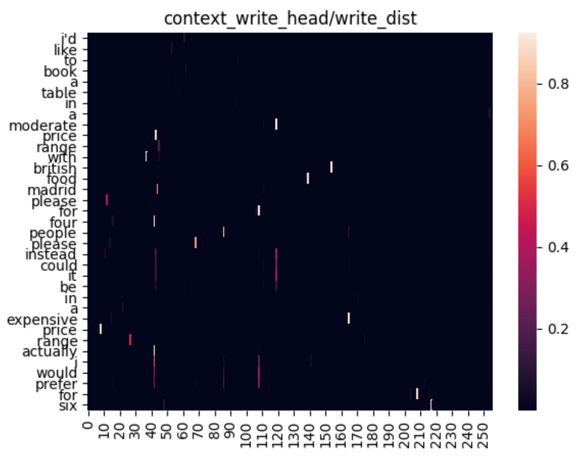

Figure 6: Visualized result of write attention weights to the context memory in the rsDNC+KM's poor performance example of dialog bAbI tasks

4.3 Movie Dialog tasks

The Movie Dialog dataset (Dodge et al., 2016) is a set of four dialogue tasks on the topic of movies. We used Task 3 (QA+Recommendation Dialog) which combines the question answering and recommendation tasks. The dialogues consist of three turns: the first turn requires a recommendation, e.g. "I'm looking for a Brian De Palma movie. Response: Blow Out", in the second turn, the user asks a factoid question regarding the model's previous response, e.g. "Who does that star? Response: John Travolta, John Lithgow, Nancy Allen, Dennis Franz", and in the third turn , the user asks for another recommendation and gives extra information about their tastes, e.g. "I prefer Robert De Niro movies. Can you suggest an alternative? Response: Hi Mom!". The dataset contains 1M examples of dialogues for training and 10k for development and test respectively. Among them, we used 100k for training and 4,907 for development, and 4,766 for test due to the limitation of computational resources. The Movie Dialog dataset's KB is built from the Open Movie Database (OMDb)[2] and the MovieLens dataset[3]. We extracted only triples sharing their entities with entities that appeared in our reduced dialogue data from the original KB, and got 126,999 triples. The number of entities is 37,055, the number of relations is 10, and the vocabulary size of the dialogues is 26,314.

Results Table 5 shows the mean hits@1 and hits@10 over three different runs of multiple models and our proposed models. Though our DNC+KM and DNC-MD+KM were worse than their corresponding original models, our rs-

[2]http://beforethecode.com/projects/omdb/download.aspx.
[3]http://grouplens.org/datasets/movielens/

Task	**DNC**	**DNC+KM**	**rsDNC**	**rsDNC+KM**	**DNC-DMS**	**DNC-DMS +KM**	**DNC-MD**	**DNC-MD +KM**
Whole test set (k=1)	**18.74%**	18.54%	18.26%	**18.54%**	18.57%	**18.61%**	**18.76%**	18.58%
Whole test set (k=10)	**37.72%**	37.49%	35.72%	**36.13%**	37.53%	**37.88%**	**38.38%**	37.33%

Table 5: Mean hits@k of different models and our proposed models in the Movie Dialog dataset Task 3 (QA+Recommendation Dialog). "k=1" and "k=10" mean hits@1 and hits@10, respectively

Task	**DNC**	**DNC+KM**	**rsDNC**	**rsDNC+KM**	**DNC-DMS**	**DNC-DMS +KM**	**DNC-MD**	**DNC-MD +KM**
Response 1 (Recs) (k=1)	**22.07%**	21.97%	14.46%	**14.56%**	21.49%	**21.95%**	**21.87%**	21.64%
Response 2 (QA) (k=1)	**7.18%**	6.80%	8.97%	**9.28%**	**7.07%**	6.82%	**7.27%**	6.87%
Response 3 (Similar) (k=1)	38.16%	**38.23%**	40.99%	**41.39%**	38.35%	**38.50%**	38.30%	**38.64%**
Response 1 (Recs) (k=10)	50.25%	**50.52%**	37.20%	**37.97%**	49.05%	**50.28%**	**50.43%**	49.36%
Response 2 (QA) (k=10)	**18.84%**	18.26%	20.62%	**20.72%**	18.79%	**18.88%**	**19.89%**	18.52%
Response 3 (Similar) (k=10)	**61.69%**	61.62%	64.28%	**64.90%**	**62.31%**	62.23%	**62.08%**	61.68%

Table 6: Detailed results in the Movie Dialog dataset Task 3 (QA+Recommendation Dialog). "k=1" and "k=10" mean hits@1 and hits@10, respectively

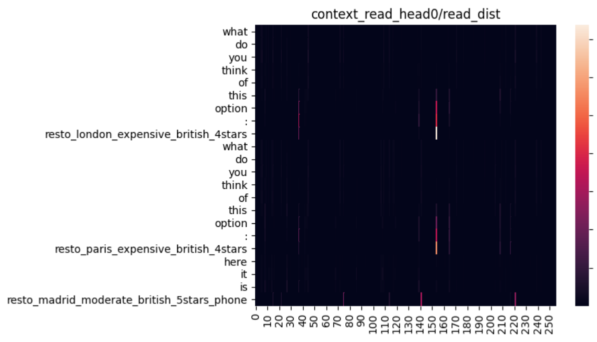

Figure 7: Visualized result of read attention weights from the context memory in the rsDNC+KM's poor performance example of dialog bAbI tasks

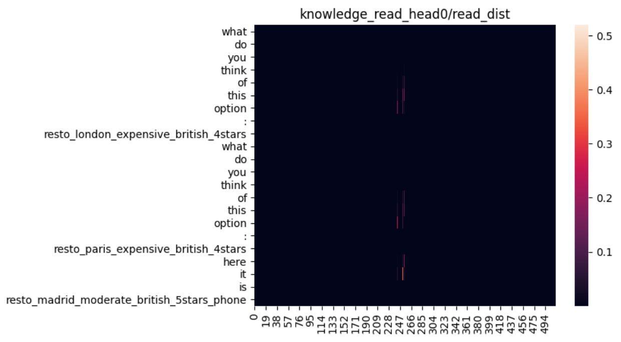

Figure 8: Visualized result of read attention weights from the knowledge memory in the rsDNC+KM's poor performance example of dialog bAbI tasks

DNC+KM and DNC-DMS+KM performed better than their original models on both hits@1 and hits@10. In Table 6, we provide the detailed results for a more specific analysis. Each task's outputs are a list of KB entities. Response 1 (Recs) is the first turn in the dialogue and requires a recommendation. The number of entities involved in response 1's test set is 5,421. Response 2 (QA) denotes the second turn and the model needs to answer factoid questions considering context from the previous turn. Response 2 has 9,867 entities in the test set since it is often asked to answer more than one entities. Response 3 (Similar) is the third turn where the model provides another recommendation given the user's extra information about their tastes. 4,939 entities are contained in the test set. In response 3 tasks, the hits@1 score of our model with the knowledge memory was higher in every DNC model. In rsDNC models, both hits@1 and hits@10 of rsDNC+KM improved on all three tasks. Despite the fact that the scores of rsDNC+KM outperformed the other models on response 2 and response 3 tasks, the results on response 1 tasks were rather low. Response 1 tasks require to deal with long input sentences such as "Gentlemen of fortune, Revanche, Eternal sunshine of the spotless mind, Prometheus, Fanny and Alexander, The hurt locker, and 127 hours are films I really like. I'm looking for a Brian De Palma movie. " and therefore we think that rsDNC models have difficulties at processing long sequences.

5 Related Work

Rae et al. (2016) introduced the sparse DNC (SDNC) with a sparse memory access scheme called Sparse Access Memory (SAM). SAM controls memory modifications within a sparse subset and uses efficient data structures for content-based addressing, and therefore the memory size does not influence memory consumption. Ben-Ari and

Bekker (2017) proposed a differentiable allocation mechanism to replace the non-differentiable sorting function of DNC and reduced training time. As different approaches to neural networks with memories, the dynamic memory network (DMN) (Kumar et al., 2015) and the DMN+ (Xiong et al., 2016), and End-to-end memory networks (Sukhbaatar et al., 2015) can be mainly listed. They store sentences in a memory and look up related sentences to answer queries using the attention mechanism. The relation memory network (RMN) (Moon et al., 2018) uses MLP and makes a multi-hop approach to an external memory to find out relevant information. In contrast to the above models, our model explicitly incorporates a memory architecture to store structured knowledge.

Key-Value Memory Networks (Miller et al., 2016) are based on End-to-end memory networks (Sukhbaatar et al., 2015) and operate a memory with the key-value structure. This structure makes the model flexible to encode knowledge sources and solves the gap between reading documents and using the KB. Saha et al. (2018) created Complex Sequential Question Answering (CSQA) dataset that consists of coherently linked questions which can be answered from a large scale KB. They combined the hierarchical recurrent encoder decoder (HRED) model (Serban et al., 2015) and the key-value memory network model (Miller et al., 2016) to solve their CSQA dataset. Unlike these models, our proposed model does not need to extract KB facts related to queries beforehand and can learn which KB facts the model should extract in a differentiable way.

6 Conclusion

We added knowledge memory architecture to three DNC models, vanilla DNC, rsDNC, and DNC-DMS, and experimentally analyzed the effect of our addition on dialogue tasks that require background knowledge. Our proposed models, DNC+KM, rsDNC+KM, and DNC-MD+KM outperformed their original models on full dialog tasks in the (6) dialog bAbI tasks dataset. In particular, each model obtained an improvement of approximately 14%, 20%, and 7%, respectively on tasks which require KB facts. In the Movie Dialog dataset, our rsDNC+KM and DNC-DMS+KM performed better than their original models. In future work we will investigate the behavior of the knowledge memory in detail and study how to build and use the knowledge memory more effectively in the whole architecture. We will also conduct experiments with models that are different from DNC models such as Key-Value Memory Networks (Miller et al., 2016) to compare with our models.

Acknowledgement

This work was supported by Grant-in-Aid for JSPS Fellows Grant Number 20J23182.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Itamar Ben-Ari and Alan Joseph Bekker. 2017. Differentiable memory allocation mechanism for neural computing.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

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

Róbert Csordás and Jürgen Schmidhuber. 2019. Improving differentiable neural computers through memory masking, de-allocation, and link distribution sharpness control. *CoRR*, abs/1904.10278.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. abs/1511.06931.

Jörg Franke, Jan Niehues, and Alex Waibel. 2018. Robust and scalable differentiable neural computer for question answering. *CoRR*, abs/1807.02658.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. Cite arxiv:1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *CoRR*, abs/1606.03126.

Jihyung Moon, Hyochang Yang, and Sungzoon Cho. 2018. Finding remo (related memory object): A simple neural architecture for text based reasoning. *CoRR*, abs/1801.08459.

John von Neumann. 1945. First draft of a report on the edvac. Technical report.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Jack W. Rae, Jonathan J. Hunt, Tim Harley, Ivo Danihelka, Andrew W. Senior, Greg Wayne, Alex Graves, and Timothy P. Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. *CoRR*, abs/1610.09027.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

H.T. Siegelmann and E.D. Sontag. 1995. On the computational power of neural nets. *J. Comput. Syst. Sci.*, 50(1):132–150.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

T Tieleman and G Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *CoRR*, abs/1603.01417.