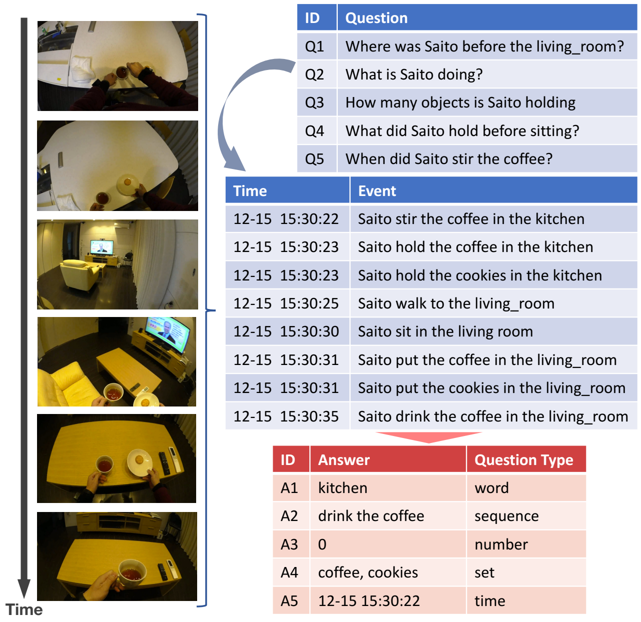
Answering Mixed-Type Questions about Daily Living Episodes

Taiki Miyanishi[1], **Jun-ichiro Hirayama,**[2,1] **Atsunori Kanemura**[3,1], **Motoaki Kawanabe**[1,2]
[1]Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan
[2]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan
[3]National Institute of Advanced Industrial Science and Technology (AIST), Ibaraki, Japan
{miyanishi, hirayama, atsu-kan, kawanabe}@atr.jp

Abstract

We propose a physical-world question-answering (QA) method, where the system answers a text question about the physical-world by searching a given sequence of sentences about daily-life episodes. To address various information needs in a physical-world situation, the physical-world QA methods have to generate mixed-type responses (e.g. word sequence, word set, number, and time as well as a single word) according to the content of questions, after reading physical-world event stories. Most existing methods only provide words or choose answers from multiple candidates. In this paper, we use multiple decoders to generate a mixed-type answer encoding daily episodes with a memory architecture that can capture short- and long-term event dependencies. Results using house-activity stories show that the use of multiple decoders with memory components is effective for answering various physical-world QA questions.



ID	Question
Q1	Where was Saito before the living_room?
Q2	What is Saito doing?
Q3	How many objects is Saito holding
Q4	What did Saito hold before sitting?
Q5	When did Saito stir the coffee?

Time	Event
12-15 15:30:22	Saito stir the coffee in the kitchen
12-15 15:30:23	Saito hold the coffee in the kitchen
12-15 15:30:23	Saito hold the cookies in the kitchen
12-15 15:30:25	Saito walk to the living_room
12-15 15:30:30	Saito sit in the living room
12-15 15:30:31	Saito put the coffee in the living_room
12-15 15:30:31	Saito put the cookies in the living_room
12-15 15:30:35	Saito drink the coffee in the living_room

ID	Answer	Question Type
A1	kitchen	word
A2	drink the coffee	sequence
A3	0	number
A4	coffee, cookies	set
A5	12-15 15:30:22	time

Figure 1: Illustration of physical-world question-answering tasks.

1 Introduction

Recording human activities in the physical-world as linguistic descriptions [Krishna *et al.*, 2017; Miyanishi *et al.*, 2018] can bring a better understanding of our everyday lives. Answering questions about such human daily episodes as in a personal digital store [Gemmell *et al.*, 2002] is an essential skill for intelligent machines that support people in their everyday activities such as human-memory aid and healthcare monitoring.

In this paper, we tackle a physical-world question-answering (QA) task where the QA method returns various answers in response to the content of a question when a sequence of sentences about physical-world events is given as in Figure 1. To achieve this scenario, the QA method requires complex reasoning referring to multiple contexts because everyday life consists of multiple events (e.g. drinking coffee in the living room after making it in the kitchen and moving to the living room.) Moreover, since people have a variety of information needs in physical-world situations, the methods of the physical-world QA tasks have to answer mixed-type responses (e.g. word sequence, word set, number, and time, as well as a single word) as to a user question. However, most existing QA methods read text and then answer questions by generating text response, extracting words from source text, or choosing from multiple candidates for answers [Richardson *et al.*, 2013; Yang *et al.*, 2015; Hermann *et al.*, 2015; Weston *et al.*, 2016; Rajpurkar *et al.*, 2016; Hill *et al.*, 2016].

Motivated by these issues, we propose a novel physical-world QA method that encodes multiple sentences describing physical-world events using Memory Networks [Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015] for capturing short- and long-term relationships and decodes mixed-type answers with multiple decoders for simultaneously learning classification, regression, sequence generation and time prediction problems. Moreover, to mitigate the different scale of each decoder loss, we incorporate self-paced learning into such multi-task learning of multiple decoders. We assume that starting with learning easy QA tasks and ending with learning relatively difficult QA tasks is effective for training multiple decoders that will output mixed-type answers.

To evaluate our physical-world QA method, we use non-synthetic everyday life stories describing various daily activities that subjects perform in a house-hold setting. The exper-

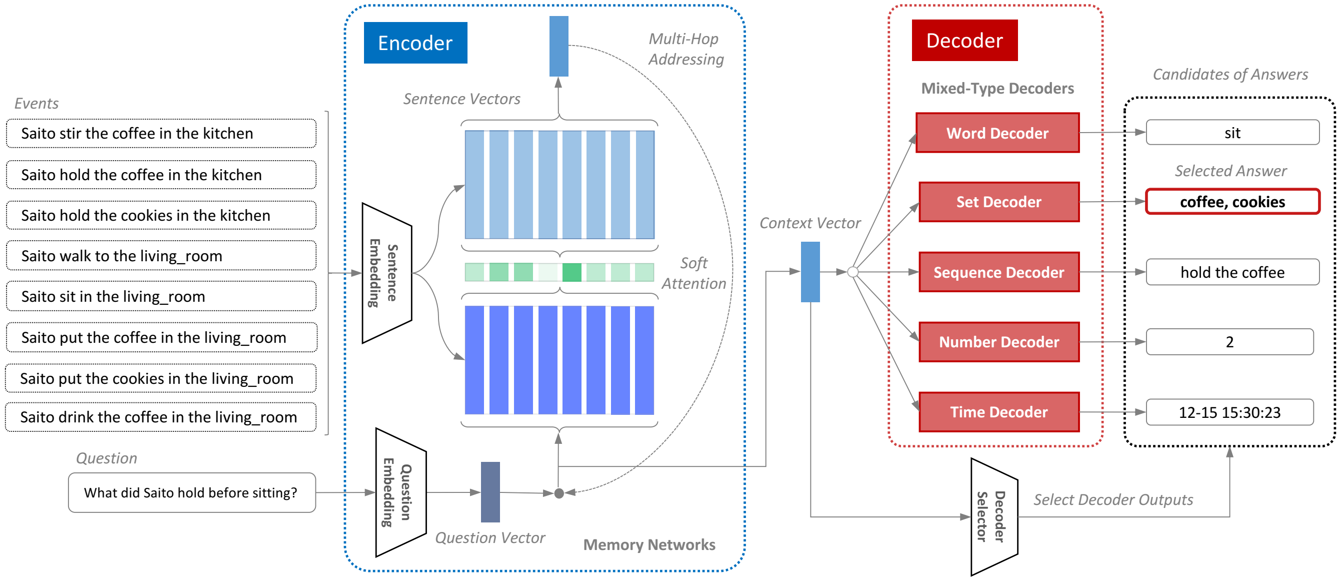


Figure 2: Illustration of proposed encoder-decoders framework.

imental results suggest that encoding physical-world events with memory components significantly improves QA performance. Moreover, we found that multi-task learning on different type QA problems is highly effective for solving physical-world QA problems and that self-paced learning of multiple decoders supports solving difficult QA tasks.

The remainder of this paper is organized as follows. First, we provide some background in question-answering for text stories. Then, we present the physical-world QA task and the proposed method to solve it. Finally, we show an experimental evaluation of daily-activity datasets collected in house-like settings and then present our conclusions.

2 Related Work

The physical-world QA can be seen as returning text response after reading a text describing a story of arbitrary length. There have been many QA tasks answering questions for stories, such as selecting an answer from multiple choices after reading materials [Richardson *et al.*, 2013; Weston *et al.*, 2016], selecting the span of text that answers a question from a large collection of documents [Yang *et al.*, 2015; Rajpurkar *et al.*, 2016], and predicting the missing named entities in a given passage [Hermann *et al.*, 2015; Hill *et al.*, 2016].

Among these tasks, the *bAbI* [Weston *et al.*, 2016][1] is closely related to our physical-world QA tasks. The dataset is commonly used for checking algorithms of text-story understanding and reasoning. In this dataset, a series of triplets (short stories, question, and answer) generated by a simple simulation similar to the approach of [Bordes *et al.*, 2010] are given. For example, when the story and a question, "Mary moved to the bathroom. John went to the hallway. Where is Mary?" is given, QA methods have to give the correct answer "bathroom". To solve this task, many deep learning methods have been proposed such as Memory Networks [Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015], Dynamic Memory Networks [Kumar *et al.*, 2016], and Differentiable Neural Computers [Graves *et al.*, 2016]. These neural-network models have an external memory for encoding a given sequence of sentences and access it using an attention mechanism [Bahdanau *et al.*, 2015] for selecting a relevant answer. The results of these works indicate using external memory as encoders is highly effective for answering questions that require solving long-term sentence dependencies compared to RNN-based models [Hochreiter and Schmidhuber, 1997; Sukhbaatar *et al.*, 2015] that output an answer after reading a sequence of words in a question and story as input. Even though we also use Memory Networks for encoding a sequence of sentences that represent daily episodes, the effectiveness of memory components is unknown when answering physical-world QA tasks. In addition, we use multiple decoders and simultaneously learn them for producing mixed-type answers that are required to respond to physical-world questions instead of selecting prepared candidates of answers or producing words or word sequences with a single decoder. Furthermore, the physical-world QA dataset we use is not artificial data created by simulation unlike the bAbI dataset. Our dataset is manually annotated by humans watching first-person videos captured by wearable cameras attached to subjects who perform various activities in a real house.

3 Proposed Method

In this section, we present our method. First, we give an overview of the proposed physical-world QA framework along with QA examples. We then proceed to the details of our QA framework based on the encoder-decoder model.

3.1 Physical-world Question Answering

Our goal is to answer questions about daily episodes in response to a given language question and temporally ordered

[1] https://github.com/facebook/bAbI-tasks

sentences that describe activities of daily living (i.e. event timeline). We use the encoder-decoder models that encode a given sequence of sentences with timestamps into internal representations as external-memory components and then decode mixed-type answers in response to a given question using multiple decoders and a decode selector.

Figure 2 shows an example of the proposed method with the event timeline when moving to the living room after preparing coffee and cookies in kitchen. Our framework answers various kinds of questions according to the content of the question. For example, the method has to answer the objects a person has when a question "What did the person hold?" is given. In addition, the method has to return a timestamp when a question "When did the person do some action".

These problems need to consider reasoning multiple facts in a sequence of sentences. To this end, we use Memory Networks as an encoder, which is a simple neural network with an external memory. Moreover, to generate mixed-type answers, we use several decoders that can solve multi-class and multi-label classification, regression, sequence generation, and time prediction problems. We introduce the encoder and the decoders in the following sections.

3.2 Memory Networks Encoder

We use Memory Networks [Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015] for encoding multiple sentences that describe physical-world states. Our Memory Networks takes a discrete set of events $\boldsymbol{e} = [e_1, \ldots, e_{|\boldsymbol{e}|}]$ that is to be stored in the memory, a question q, and outputs an answer $\boldsymbol{a}$, where q has timestamp q^t when the question was issued, e has timestamp e^t, which is a start-time of an event when a subject does ADL (activity of daily living), and $\boldsymbol{a}$ is a tuple $(a_1, \ldots, a_{|\boldsymbol{a}|})$. Note that $\boldsymbol{a}$ becomes mixed-type data in response to the content of a question. For example, $\boldsymbol{a}$ denotes a number $a = 3$ when predicting how many objects he/she is holding. Each of the e_i and q contains symbols coming from a dictionary with V words. Each answer $\boldsymbol{a}$ also comes from a dictionary with V words when predicting a word, word sequence and set.

Content-Based Memory Addressing

We convert all events e_i into N real-valued vectors $\boldsymbol{m}_i$ and $\boldsymbol{m'}_i$ for input and output of memory representations. Old events are discarded when $|\boldsymbol{e}| > M$, where M is the memory size. We compute each $\boldsymbol{m}_i$ and $\boldsymbol{m'}_i$ by embedding each e_i in a continuous space using embedding matrices $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{R}^{N \times V}$, respectively. We also embed the question q with another embedding matrix $\boldsymbol{B} \in \mathbb{R}^{N \times V}$ to obtain an internal state $\boldsymbol{u}$. To read memory in response to question q, we use soft attention defined as follows:

$$\boldsymbol{o} = \sum_{i=1}^{K} \boldsymbol{p}_i \boldsymbol{m'}_i, \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^N$ is a vector of non-negative weights that sum to one. Each weight $\boldsymbol{p}$ is used for addressing content in memory related to a question. We define for the i-th read weight,

$$\boldsymbol{p}_i = \mathrm{Softmax}(\boldsymbol{u}^T \boldsymbol{m})_i \tag{2}$$

where $\mathrm{Softmax}(\boldsymbol{z})_i = \exp(\boldsymbol{z}_i) / \sum_j \exp(\boldsymbol{z}_j)$. The weight $\boldsymbol{p}_i$ can be seen as a probability vector over the inputs. We compute gradients in all rows of memory per time step.

Multi-Hop Addressing

We stack memory layers to handle l hop operations in Memory Networks. The input to layers above the first is the sum of the output $\boldsymbol{o}_k$ and the input $\boldsymbol{u}_k$ from layer k:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{o}_k. \tag{3}$$

Each layer has its own embedding matrices $\boldsymbol{A}_k$, $\boldsymbol{C}_k$, used to embed the inputs. To ease training and reduce the number of parameters, we use adjacent weight. Thus, the output embedding for one layer is the input embedding for the one above, i.e. $\boldsymbol{A}_{k+1} = \boldsymbol{C}_k$. In addition, we constrain the question embedding to match the input embedding of the first layer, i.e. $\boldsymbol{B} = \boldsymbol{A}_1$. After memory addressing l times, we get the input embedding $\boldsymbol{u}_{l+1}$ that is a context vector $\boldsymbol{c} \in \mathbb{R}^N$.

3.3 Mixed-Type Decoders

To respond to mixed-type answers, we use multiple decoders that transform the context vector $\boldsymbol{c}$ into mixed-type data such as word, word set, word sequence, number, and time (memory pointer). We also use a decode selector for selecting an optimal decoder suitable for each problem as in Figure 2. By using decode selectors, the framework selects decode outputs in response to the content of question. An index of the decoder for the final prediction is

$$i^* = \operatorname*{argmax}_{i=1,\ldots,D} \mathrm{Softmax}(\boldsymbol{W}^d \boldsymbol{c}) \tag{4}$$

where $\boldsymbol{W}^d \in \mathbb{R}^{D \times N}$ and D is the number of decoders. The final prediction is

$$\boldsymbol{a} = \mathrm{Decoder}_{i^*}(\boldsymbol{c}), \tag{5}$$

where $\mathrm{Decoder}_i(\cdot)$ outputs i-th decoder. The parameter of the decode selector is updated by the cross-entropy between its outputs and the question-type label attached to each question-answer pair in the training data. We introduce each decoder in the following sections.

Word Decoder

To predict a single word as answer, we use the multi-class classification framework. The method predicts a label $\boldsymbol{a} = w$ from vocabulary V based on the context vector $\boldsymbol{c}$. We use the weight matrix $\boldsymbol{W}^c \in \mathbb{R}^{V \times N}$ and a softmax function to produce the indicator that selects the word.

$$\hat{\boldsymbol{w}} = \mathrm{Softmax}(\boldsymbol{W}^c \boldsymbol{c}). \tag{6}$$

We update their parameters while minimizing a cross-entropy loss between the predicted probabilities and the true answers as $\mathcal{L}_{word}(\hat{\boldsymbol{w}}, \boldsymbol{w}) = -\sum_i \boldsymbol{w}_i \log(\hat{\boldsymbol{w}}_i)$.

Set Decoder

To predict a word set in response to a question, we use a multi-labeling approach. The method predicts a tuple $\boldsymbol{a} = (w_1, \ldots, w_m)$, where m is the size of a word set and each w is from vocabulary V based on the context vector $\boldsymbol{c}$. We use the weight matrix $\boldsymbol{W}^m \in \mathbb{R}^{V \times N}$ and a softmax to produce the output of the decoder as

$$\hat{\boldsymbol{w}} = \mathrm{Sigmoid}(\boldsymbol{W}^m \boldsymbol{c}), \tag{7}$$

where $\mathrm{Sigmoid}(\boldsymbol{z}_i) = 1/(1 + \exp(-\boldsymbol{z}_i))$. We select top m words with higher values in $\hat{\boldsymbol{w}}$ as an answer. To update parameters, we use the loss function that optimizes a multi-label one-versus-all loss defined as $\mathcal{L}_{set}(\hat{\boldsymbol{w}}, \boldsymbol{w}) = -\sum_i \big\{\boldsymbol{w}_i \log(1 - \hat{\boldsymbol{w}}_i) + (1 - \boldsymbol{w}_i) \log(\hat{\boldsymbol{w}}_i)\big\}$.

Sequence Decoder

For the word-sequence generation, we use a recurrent neural network language model (RNNLM) [Mikolov *et al.*, 2010] that produces a word sequence $\boldsymbol{a} = [w_1, w_2, \ldots, w_n]$, where w from vocabulary V and n is the number of words in the answer. RNNLM updates the following equations:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{w}_t), \tag{8}$$

$$\hat{\boldsymbol{w}}_t = \mathrm{Softmax}(\boldsymbol{W}^s \boldsymbol{h}_t + \boldsymbol{b}), \tag{9}$$

where $f(\cdot)$ is a function of RNN, $\boldsymbol{W}^s \in \mathbb{R}^{V \times O}$, O is the number of hidden units (N in our case), and $\boldsymbol{b}$ is a bias term. We initialize $\boldsymbol{h}_0$ with the context vector $\boldsymbol{c}$. We use the cross entropy between $\boldsymbol{w}_{t+1}$ and $\hat{\boldsymbol{w}}_t$ over $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t\}$, then $\mathcal{L}_{seq}(\{\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_n\}, \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}) = -\sum_t \boldsymbol{w}_{t+1} \log(\hat{\boldsymbol{w}}_t)$.

Number Decoder

To predict a number, we solve the regression problem. The method predicts a label $\boldsymbol{a} = r \in \mathbb{R}$ based on the context vector $\boldsymbol{c}$. We use the weight vector $\boldsymbol{w}^r \in \mathbb{R}^N$ as

$$\hat{r} = \boldsymbol{w}^r \boldsymbol{c}, \tag{10}$$

where $y \in \mathbb{R}$. When predicting an integral number, we cast the real number into the integer as an output. We use a squared loss $\mathcal{L}_{num}(\hat{r}, r) = (\hat{r} - r)^2$ to update parameters.

Time Decoder

To solve the time prediction problem, we use a binary-class classification framework that indicate a specific index of memory (memory pointer) $s \in [1, \ldots, M]$ relevant to the answer based on the context vector $\boldsymbol{c}$. We use the weight matrix $\boldsymbol{W}^s \in \mathbb{R}^{(M+1) \times N}$ and a softmax function to produce the indicator that selects the output of the decoder:

$$\hat{\boldsymbol{s}} = \mathrm{Softmax}(\boldsymbol{W}^s \boldsymbol{c}), \tag{11}$$

where the index of $M + 1$ indicates out of memory. We update their parameters while minimizing a cross-entropy loss between the predicted probabilities and the true answers as $\mathcal{L}_{time}(\hat{\boldsymbol{s}}, \boldsymbol{s}) = -\sum_i \boldsymbol{s}_i \log(\hat{\boldsymbol{s}}_i)$. This decoder outputs an timestamp of event $\boldsymbol{a} = e^t_s$ when testing.

3.4 Self-Paced Learning for Multiple Decoders

We train the proposed encoder-decoders based on neural networks with multi-task learning, which is widely used for many natural language processing (NLP) tasks such as multilingual neural machine translation [Luong *et al.*, 2016] and a joint learning of a wide variety of NLP tasks [Collobert and Weston, 2008; Hashimoto *et al.*, 2017]. The advantage of multi-task learning of neural networks is that it can learn useful shared features for related tasks and robustly improve performance with limited data. However, each decoder in our proposed method has a different scale of loss values, since each decoder outputs a different type data.

To address this problem, we use self-paced learning for simultaneously learning our encoder-decoder model. We follow the self-paced learning for multi-task problems [Murugesan and Carbonell, 2017]. The objective function $\tilde{\mathcal{L}}$ of the self-paced learning for T tasks is given as:

$$\tilde{\mathcal{L}}(\tau, \theta, \boldsymbol{X}, \boldsymbol{y}) = \sum_{t \in T} \tau_t \mathcal{L}(\theta_t, \boldsymbol{X}_t, \boldsymbol{y}_t) + \lambda r(\tau), \tag{12}$$

where $\mathcal{L}(\theta_t, \boldsymbol{X}_t, \boldsymbol{y}_t) = \frac{1}{N_t} \sum_{i \in N_t} l(\theta^i_t, \boldsymbol{X}^i_t, \boldsymbol{y}^i_t)$. Note that each $\mathcal{L}$ corresponds to $\mathcal{L}_{word}$, $\mathcal{L}_{set}$, $\mathcal{L}_{seq}$, $\mathcal{L}_{num}$, and $\mathcal{L}_{time}$. $l(\theta^i_t, \boldsymbol{X}^i_t, \boldsymbol{y}^i_t)$ is each loss function of a i-th instance given that t-th task is associated with N_t training examples. Since our method of shared knowledge in the encoder and regularization term is included in each loss, we omit the regularization term over decoders. The learning algorithm considers a task easy if it has low training error. We define τ_t as

$$\tau_t = \begin{cases} 1 & \mathcal{L}(\theta_t, \boldsymbol{X}_t, \boldsymbol{y}_t) < \lambda \\ \delta & otherwise. \end{cases}$$

We set $\tau_t = \delta$ for the hard tasks. We also use the parameter c that controls the learning pace of the self-paced procedure (i.e., $\lambda \leftarrow c\lambda$ for each iteration). We set the c value as greater than 1 for relaxing threshold λ.

4 Experiments

In this section, we reports on the empirical evaluation for the proposed physical-world QA method on a daily activity dataset.

4.1 Dataset

We make a document collection describing various daily activities in a house-hold setting. To this end, we re-annotate a daily activity dataset that has been used in the egocentric video retrieval and the event timeline generation [Miyanishi *et al.*, 2016; 2018] for physical-world QA problems. It contains first-person videos captured by a head-mounted wearable camera attached to eight subjects. They performed 20 continuous ADLs in six different places 10 times (i.e. a total of 10 sessions) under the semi-naturalistic collection protocol [Bao and Intille, 2004] for collecting variable activities. For example, the subject makes a cookie in the kitchen, moves to the living room with it, and eats it after sitting on the sofa. Total video length is about 17 hours. We define physical-world events in one session as a single story indicating that each subject has 10 stories. To evaluate physical-world QA methods, we re-annotated this dataset with structured event descriptions of the ADLs that subjects performed in a house. We obtained 11,497 sentences of ADL events with 1,031 unique events. For QA tasks, we prepared 22 question and answer templates that ask about the state of the physical-world. Each template is categorized by question types corresponding to each decoder output. We prepared questions and their answers on the collected stories using these templates. Table 1 shows the template of questions and answers we used for generating question-and-answer pairs of each task. We randomly insert 25 questions into each story. If there are no answers to questions, an “unknown” token is used for answers. In total, our QA dataset, which we call a daily living QA dataset, has 2,000 questions for each question answer template. Figure 3 shows examples of our QA tasks.

4.2 Baselines

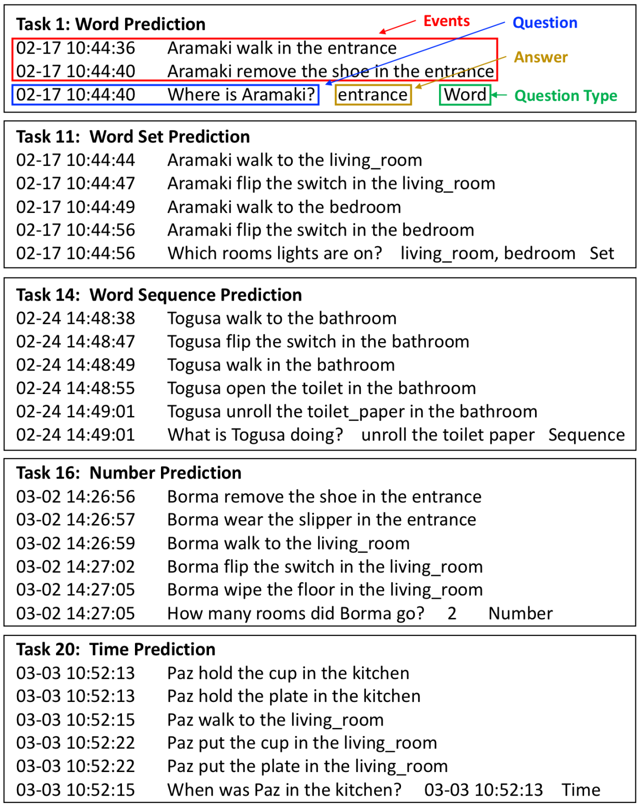
We prepared the proposed method and several baselines based on the encoder-decoder model:

Question	Answer
Word	
Where is [person]?	[place]
Where is [object]?	[place]
Did [person] go to [place]?	yes/no
Where was [person] before [place]?	[place]
Where was [person] after [place]?	[place]
Who is in [place]?	[person]
Who did [action] [object]?	[person]
Who did [action] [object] in [place]?	[person]
Where did [person] [action] [object]?	[place]
Set	
What is [person] holding?	[object]+
Which rooms lights are on?	[place]+
Which rooms did [person] go?	[place]+
Sequence	
What is [person] doing?	[action] [object]
What did [person] do after [action] [object]?	[action] [object]
What did [person] do before [action] [object]?	[action] [object]
Number	
How many rooms did [person] go?	[number]
How many rooms lights are on?	[number]
How many objects is [person] holding?	[number]
How many times did [person] [action] [object]?	[number]
Time	
When was [person] in [place]?	[time]
When did [person] [action]?	[time]
When did [person] [action] in [place]?	[time]

Table 1: Question and answer template on each QA task used for generating questions and answers, where [person], [object], and [place] are words extracted from stories, [time] is extracted from the timestamp field, and '+' indicates multiple words.

- RNN+RNN [Sutskever *et al.*, 2014] is a standard sequence-to-sequence model that encodes the story until the point it reaches a question and then decodes word sequence as an answer.
- RNN+MixDecs encodes a story with RNN and then decodes mixed-type answers in response to a question using multiple decoders.
- MemNN+RNN [Sukhbaatar *et al.*, 2015] is standard Memory Networks that encodes a story with multiple sentence embeddings and addresses them in response to a question embedding and then decodes word sequence as an answer.
- MemNN+MixDecs encodes a story and a question with Memory Networks into a context vector and decodes it to mixed-type answers using multiple decoders.
- MemNN+MixDecs+SelfPaced is equal to MemNN+MixDecs except that self-paced learning is used when learning this model.

RNN+RNN and RNN+MixDecs have a disadvantage with regard to encoding a long story compared to MemNN+RNN and MemNN+MixDecs, which use external memory. Moreover, they cannot answer questions about time since RNN simply reads the story as a sequence of words and does not distinguish each physical-world event. RNN+RNN and MemNN+RNN use RNN for generating an answer, so they cannot predict continuous numbers such as time. Thus, the use of multiple decoders enables us to use more training examples for multi-task learning. We also prepared MemNN+MixDecs+SelfPaced for whether the self-paced learning contributes our QA tasks.


Events
Question
Answer
Question Type

Task 1: Word Prediction
02-17 10:44:36 Aramaki walk in the entrance
02-17 10:44:40 Aramaki remove the shoe in the entrance
02-17 10:44:40 Where is Aramaki? entrance Word

Task 11: Word Set Prediction
02-17 10:44:44 Aramaki walk to the living_room
02-17 10:44:47 Aramaki flip the switch in the living_room
02-17 10:44:49 Aramaki walk to the bedroom
02-17 10:44:56 Aramaki flip the switch in the bedroom
02-17 10:44:56 Which rooms lights are on? living_room, bedroom Set

Task 14: Word Sequence Prediction
02-24 14:48:38 Togusa walk to the bathroom
02-24 14:48:47 Togusa flip the switch in the bathroom
02-24 14:48:49 Togusa walk in the bathroom
02-24 14:48:55 Togusa open the toilet in the bathroom
02-24 14:49:01 Togusa unroll the toilet_paper in the bathroom
02-24 14:49:01 What is Togusa doing? unroll the toilet paper Sequence

Task 16: Number Prediction
03-02 14:26:56 Borma remove the shoe in the entrance
03-02 14:26:57 Borma wear the slipper in the entrance
03-02 14:26:59 Borma walk to the living_room
03-02 14:27:02 Borma flip the switch in the living_room
03-02 14:27:05 Borma wipe the floor in the living_room
03-02 14:27:05 How many rooms did Borma go? 2 Number

Task 20: Time Prediction
03-03 10:52:13 Paz hold the cup in the kitchen
03-03 10:52:13 Paz hold the plate in the kitchen
03-03 10:52:15 Paz walk to the living_room
03-03 10:52:22 Paz put the cup in the living_room
03-03 10:52:22 Paz put the plate in the living_room
03-03 10:52:15 When was Paz in the kitchen? 03-03 10:52:13 Time

Figure 3: Examples of a physical-world QA dataset that contains physical-world events, questions, and their answers. We use alias names for subjects.

4.3 Experimental Settings

We used the QA dataset about four subjects for a training set and two subjects for a validation set, which was used to tune model- and hyper-parameters. We trained all methods using a learning rate 0.01 and a batch size of 128 until 10 epochs were reached. The weights were initialized randomly from a Gaussian distribution $\mathcal{N}(0, 0.01)$. A null symbol was used to pad them all to a fixed size. The embedding of the null symbol was constrained to be zero. For all RNN encoders and decoders, we used a gated recurrent unit (GRU) [Cho *et al.*, 2014] with two stacked hidden layers and a bidirectional RNN [Schuster and Paliwal, 1997]. For all methods, we make the dimension of the hidden states equal to the dimension of the word embeddings and select the the dimension of states from $N = \{128, 256, 512\}$. We used a dropout [Srivastava *et al.*, 2014] rate of 0.5 for the outputs of each RNN layer, and the linear transformation be-

Question Type	RNN+ RNN	RNN+ MixDecs	MemNN+ RNN	MemNN + MixDecs	MemNN+ MixDecs+ SelfPaced
Word	0.614	0.607	0.749	**0.787**	0.766
Set	0.747	0.570	0.818	0.838	**0.849**
Sequence	0.518	0.564	0.590	0.689	**0.741**
Number	0.545	0.554	0.715	0.676	**0.744**
All	0.605	0.583	0.728	0.756	**0.771**

Table 2: Physical-world QA performance on 19 QA tasks without the *Time* prediction task. Best results are bolded.

Question Type	MemNN + RNN	MemNN + MixDecs	MemNN + MixDecs + SelfPaced
Word	0.752	**0.775**	0.771
Set	0.807	**0.835**	**0.835**
Sequence	0.628	0.621	**0.707**
Number	0.735	0.706	**0.775**
Time	0.499	**0.786**	0.719
All	0.705	0.751	**0.765**

Table 3: Physical-world QA performance on 22 QA tasks. Best results are bolded.

fore the softmax for the answers except for the number decoder. For MemNN+RNN and MemNN+MixDecs, we set a fixed number ($l = 3$ hops) in the encoding of Memory Networks. For MemNN+MixDecs, we also used self-paced learning selecting parameter candidates among $\lambda = \{0.1, 1\}$, $\delta = \{0.01, 0.1\}$, and $c = \{1.01, 1.1\}$. We evaluated all methods under the ParlAI framework [Miller *et al.*, 2017] and used F1 measure as an evaluation metric. We separately trained and predicted when the question type is *Time* or not since the RNN encoder does not have explicit memory to point out a timestamp. Note that we simultaneously learn multiple decoders unless otherwise noted.

4.4 Results

In this section, we describe the experimental results over all methods when using the daily living QA dataset. Table 2 shows the QA performances across 19 tasks except for the *Time* question type using RNN+RNN, RNN+MixDecs, MemNN+RNN, MemNN+MixDecs. The overall results showed that MemNN+RNN and MemNN+MixDecs outperformed RNN+RNN and RNN+MixDecs respectively indicating that memory components mounted on Memory Networks significantly improve the QA performance on daily episodes, which requires short-term and long-term dependencies in QA compared to RNN. These results are similar to using the simulation dataset introduced in past work [Sukhbaatar *et al.*, 2015]. With regard to decoder performance, MemNN+MixDecs outperformed MemNN+RNN on average when used with and without time questions. The results suggest that using multiple decoders with memory components is effective for a wide variety of questions compared to decodes with RNN that generate only word sequence. Table 3 shows the QA performances across all 22 tasks. The results show that MemNN+MixDecs+SelfPaced outperformed MemNN+MixDecs, suggesting that weighting decoders with self-paced learning is effective when learning multiple decoders for our QA tasks.

Question Type	MemNN + MixDecs (Each)	MemNN + MixDecs (Joint)	MemNN + MixDecs + SelfPaced (Joint)
Word	**0.776**	0.775	0.771
Set	0.822	**0.835**	0.835
Sequence	0.507	0.621	**0.707**
Number	0.576	0.706	**0.775**
Time	0.610	**0.786**	0.719
All	0.687	0.751	**0.765**

Table 4: Physical-world QA performance of learning by each question type or joint learning of various questions. Best results are bolded.

Table 2 also shows the QA performances on each question type except for the *Time* question type. Across all types of answers, MemNN+RNN and MemNN+MixDecs outperformed RNN+MixDecs and RNN+RNN. The results indicate that memory components are essential for improving QA performance for all types of question. From the results in table *Sequence* and *Number*, questions are relatively difficult. MemNN+MixDecs+SelfPaced improved these questions. We assumed that self-paced learning, which learns more difficult problems after first learning simple problems, improves this performance. The results of MemNN+MixDecs+SelfPaced outperforming MemNN+MixDecs support this idea.

Table 4 shows the results of learning the encoder-decoder model by each question type or joint learning of various questions. We assumed that jointly learning multiple problems despite using mixed-type decoders is good for solving various questions since this method can learn useful features in the shared encoder for related tasks augmenting training data and then improve its performance. Comparing MemNN+MixDecs (Each) to MemNN+MixDecs (Joint) and MemNN+MixDecs+SelfPaced (Joint), the results show that joint learning is effective for improving most question types. The results indicates that the multi-task learning on different type QA problems is highly effective for solving physical-world QA problems.

5 Conclusion

We have shown that physical-world QA tasks answer various questions by reading physical-world events in daily episodes. To solve this problem, we proposed an encoder-decoder model combining memory components and multi-task learning for mixed-type decoders. The experimental results of using a daily living QA dataset indicate that multi-task learning of several different QA problems is suitable for solving problems that are important for the physical-world QA tasks.

Acknowledgments

This work was supported by JST CREST Grant Number JP-MJCR15E2 and JSPS KAKENHI Grant Number 16K21718.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[Bao and Intille, 2004] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive*, pages 1–17, 2004.

[Bordes *et al.*, 2010] Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. Towards understanding situated natural language. In *AISTATS*, 2010.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[Gemmell *et al.*, 2002] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. Mylifebits: Fulfilling the memex vision. In *ACMMM*, pages 235–238, 2002.

[Graves *et al.*, 2016] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[Hashimoto *et al.*, 2017] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *EMNLP*, 2017.

[Hermann *et al.*, 2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, 2015.

[Hill *et al.*, 2016] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*, 2016.

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

[Krishna *et al.*, 2017] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *ICCV*, pages 706–715, 2017.

[Kumar *et al.*, 2016] 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 *ICML*, volume 48, pages 1378–1387, 2016.

[Luong *et al.*, 2016] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *ICLR*, 2016.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[Miller *et al.*, 2017] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *EMNLP System Demonstrations*, pages 79–94, 2017.

[Miyanishi *et al.*, 2016] Taiki Miyanishi, Jun-ichiro Hirayama, Takuya Maekawa, Quan Kong, Hiroki Moriya, and Takayuki Suyama. Egocentric video search via physical interactions. In *AAAI*, pages 330–336, 2016.

[Miyanishi *et al.*, 2018] Taiki Miyanishi, Jun-ichiro Hirayama, Takuya Maekawa, and Motoaki Kawanabe. Generating an event timeline about daily activities from a semantic concept stream. In *AAAI*, 2018.

[Murugesan and Carbonell, 2017] Keerthiram Murugesan and Jaime Carbonell. Self-paced multitask learning with shared knowledge. In *IJCAI*, pages 2522–2528, 2017.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, 2016.

[Richardson *et al.*, 2013] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203, 2013.

[Schuster and Paliwal, 1997] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process*, 45(11):2673–2681, 1997.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[Sukhbaatar *et al.*, 2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[Weston *et al.*, 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *ICLR*, 2015.

[Weston *et al.*, 2016] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *ICLR*, 2016.

[Yang *et al.*, 2015] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018, 2015.