Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Sim2RealQA: Using Life Simulation to Solve Question Answering Real-world Events

TAIKI MIYANISHI[1,3], TAKUYA MAEKAWA [2], AND MOTOAKI KAWANABE.[1,3]
[1]Advanced Telecommunications Research Institute International (ATR), 2-2-2 Hikaridai, Seika Town, Kyoto 619-0288, Japan
[2]Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
[3]RIKEN Center for Advanced Intelligence Project (RIKEN AIP), 2-2-2 Hikaridai, Seika Town, Kyoto 619-0288, Japan

Corresponding author: Taiki Miyanishi (e-mail: miyanishi@atr.jp).

ABSTRACT As smart speakers continue to proliferate, question answering (QA) by smart devices is being woven into our daily lives. This study assumes question answering related to daily life events detected by context recognition systems, such as activity recognition and indoor positioning systems, e.g., answering questions like “Did my grandma eat dinner?” and “How many times did my grandpa go to the toilet?” These questions can effectively support human memory-aids, locate lost items, and monitor human activities. However, training a question-answering model requires large amounts of labeled training data (i.e., questions, answers, and the time-series of real-world event triplets) collected in a target environment. In this paper, we propose a novel simulation to real QA (Sim2RealQA) framework that completely trains a QA model with QA datasets produced in a life simulator and use it for solving real-word QA problems without answer labels. Our proposed QA model can learn a general reasoning process for QA that is independent of environments and deal with diverse types of questions specific to question answering in real-world environments, e.g., counting the number of occurrences of a real-world event and enumerating the names of those who are performing an activity together. Experiments show that using life simulations is a promising approach for solving real-world QA problems when no real-world answer labels are available.

INDEX TERMS Lifelog, ubiquitous computing, daily-living activities, real-world question answering.

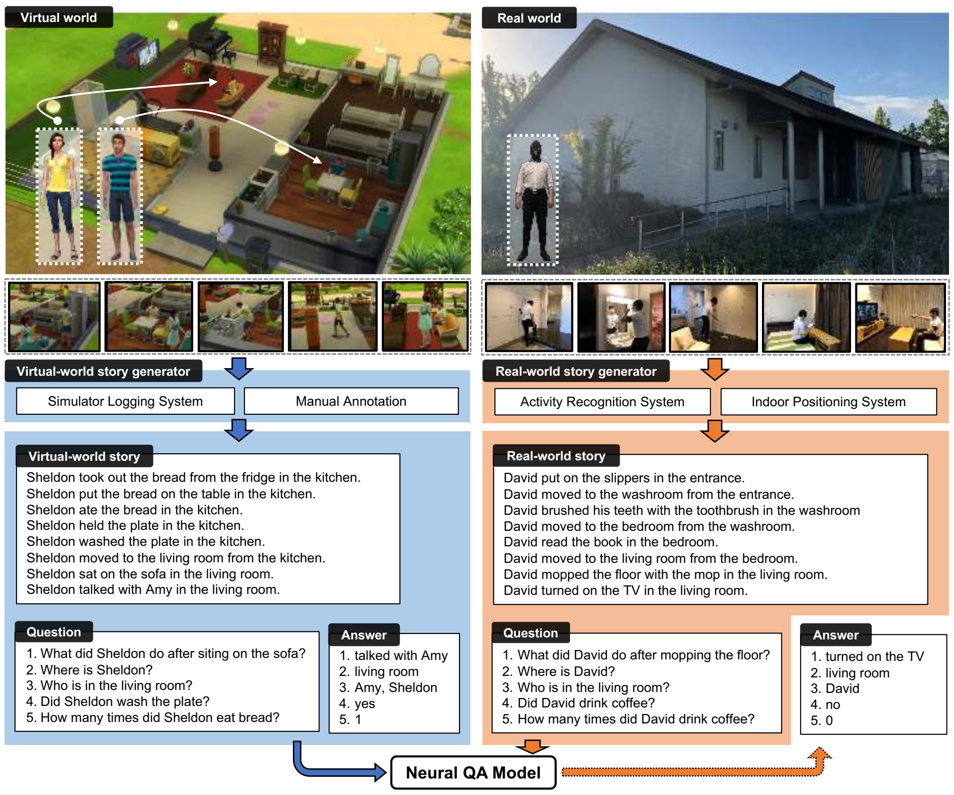
I. INTRODUCTION

A. BACKGROUND

Based on the recent advances in sensing methods, such context recognition technologies as activity recognition and indoor positioning have been scrutinized in the IoT community. Many activity recognition studies employ body-worn sensors, including acceleration sensors, gyroscopes, cameras, and microphones to recognize such daily activities as walking, running, and house cleaning [1]–[5]. Indoor positioning studies rely on signaling technologies, for example, infrared [6], ultrasound [7], active sound probing [8], [9], Bluetooth [10], and Wi-Fi [11], [12]. The recognized context information can be used in real-world services, e.g., context-aware systems, lifelogging, and the surveillance of the elderly [13]–[17].

Due to the recent proliferation of such smart speakers as Amazon Echo and Google Home, question answering (QA) by these smart devices is being woven into our daily lives. As mentioned above, our daily lives are being monitored by such context recognition techniques as activity recognition and indoor positioning. Based on the recognized and stored daily activity data, real-world question answering (real-world QA) has been investigated [18]. Real-world QA, which provides more fine-grained understanding of our daily living than just retrieving past events, offers many useful real-world applications for improving quality of life. For instance, answering such questions as “What did I eat last night?,” “Where is my smartphone?,” and “Did Mary take her medicine after eating?” supports human memory-aids, locates lost items, and monitors human activities.

Because a real-world QA is assumed to output a linguistic description as an answer, which is read out by a smart speaker, recognized daily life events are stored as linguistic descriptions (a series of sentences related to real-world events) that facilitate answer generation from the stored sentences related to daily life based on the state-of-the-art QA methods. For example, when an indoor positioning system detects that Mary’s current indoor coordinates have changed



Sheldon took out the bread from the fridge in the kitchen.
Sheldon put the bread on the table in the kitchen.
Sheldon ate the bread in the kitchen.
Sheldon held the plate in the kitchen.
Sheldon washed the plate in the kitchen.
Sheldon moved to the living room from the kitchen.
Sheldon sat on the sofa in the living room.
Sheldon talked with Amy in the living room.

1. What did Sheldon do after siting on the sofa?
2. Where is Sheldon?
3. Who is in the living room?
4. Did Sheldon wash the plate?
5. How many times did Sheldon eat bread?

1. talked with Amy
2. living room
3. Amy, Sheldon
4. yes
5. 1

David put on the slippers in the entrance.
David moved to the washroom from the entrance.
David brushed his teeth with the toothbrush in the washroom
David moved to the bedroom from the washroom.
David read the book in the bedroom.
David moved to the living room from the bedroom.
David mopped the floor with the mop in the living room.
David turned on the TV in the living room.

1. What did David do after mopping the floor?
2. Where is David?
3. Who is in the living room?
4. Did David drink coffee?
5. How many times did David drink coffee?

1. turned on the TV
2. living room
3. David
4. no
5. 0

FIGURE 1. Sim2RealQA example: We train models with a virtual-world QA dataset produced in a life simulator (**left**) to solve real-world QA problems (**right**).

from the living room to the kitchen, this event is converted to the following sentence: "Mary has moved from the living room to the kitchen." Since real-world QA requires complex reasoning and the generation of diverse answers (e.g., numbers, "yes/no," and a sequence of words) in response to various information needs in the real-world, the current real-world QA approaches mainly use neural network-based QA models that demonstrate high performance over many story-based QA tasks [19]–[21]. However, neural network-based model performance relies heavily on large training datasets [22], [23].

In real-world QA, question and answer pairs as well as sentences about daily stories collected in a target environment are required as a training dataset. 5,000 QA pairs and 1,500 sentences are required for each environment [18]. However, preparing a sufficient amount of real-world QA dataset is costly and impractical. Preparing QA pairs is especially difficult because daily life events must be observed by someone to answer real-world questions, which also raises severe privacy concerns.

B. APPROACH

To address these problems, we propose to use a life simulator to produce sufficient amounts of QA datasets for training neural QA models. With a life simulator, we can easily create realistic daily living environments rather than building actual houses and obtain diverse realistic daily life stories. With virtual daily life stories, a large amount of virtual-world QA datasets can be efficiently compiled without breaching privacy concerns. Due to such advantages, we trained a neural QA model with this virtual-world QA dataset and solved real-world QA problems without real-world labeled data. We designated this proposed framework as a simulation to a real QA (*Sim2RealQA*). Fig. 1 presents an example of our proposed framework. In this study, we use a life-simulator game[1] (e.g., The Sims), which replicates a person's life in a virtual world. Because a person performs a variety of activities while interacting with common objects and others to replicate real-world daily life activities, we can generate a sufficient amount of information about daily events to acquire reasoning rules of QA in normal daily lives regardless of privacy issues.

We assume the existing activity recognition and indoor positioning methods studied in the IoT community and generate a sequence of sentences about daily stories in a real-world environment based on the expected outputs of these methods. We generate sentences by embedding the information acquired by the activity recognition and indoor positioning methods into a template, such as "Subject + Predicate + Object + Location." Assume that the activity recognition system detects that David has started a "sleeping" activity. In addition, an indoor positioning system has tracked his position, and his location is estimated to be a bedroom. By referring to the predictions, "David slept in the bed-

[1]In some of the figures of this paper, we cite screenshots of gameplay.

room" is generated for that event. In this study, we prepared templates that can decode the results of various types of activity recognition and indoor positioning methods. Based on these templates, we generated a sequence of sentences about daily stories in real/virtual environments. We believe that generating sentences about daily life stories from sensor data has the following benefits. (i) We can design our real-world QA model based on state-of-the-art QA techniques in NLP studies, which assume linguistic descriptions as input. (ii) Because a question (and an answer read out by a smart speaker) is done by natural language, QA targets (i.e., events) that have the same modality as the question are easy to find. (iii) Since various activity recognition and indoor positioning systems are available, the output formats of such systems also vary. When outputs are stored in a relational database, for example, the columns of tables depend on activity recognition/indoor positioning systems. Therefore, a process that handles information from the systems greatly depends on the output formats (e.g., table structures) and must be hand-crafted for each system, complicating the implementation of these processes in a neural network model. In contrast, our approach can deal with the outputs of any type of system once templates are prepared for each one.

In addition to sentences about real-world stories, we generate typical real-world questions based on the entities (e.g., object, place, and person) in a real/virtual environment, which are used to train/evaluate a QA model. We made various types of questions that can be used in real-world situations, including "Where is Sally?", "What did Ann do before going to bed?", "Who opened the refrigerator?", and "How many times did Tom drink coffee?" The answers require different forms: "bathroom," "used her smartphone," "Ann, Sally, and Tom," and "5".

To train a QA model that is applicable for such real-world QA problems using a virtual-world QA dataset, we designed a QA model so that it learns a general reasoning process that can be used in any environment. Assume that the following time-series of sentences about daily stories (story events) are generated by activity recognition and indoor positioning systems:

1) Mary moved to the washroom from the entrance.
2) Tom drank coffee in the living room.
3) Mary washed her hands in the washroom.
4) Mary brushed her teeth in the washroom.
5) Tom moved to the kitchen from the living room.
6) Tom opened the fridge in the kitchen.

In addition, a question is asked: "Where is Mary?" To answer it, a naive QA model may calculate the output probabilities of each word in vocabulary. In this case, the probability related to "washroom" is expected to be high. However, the naive model cannot output answer words that are not included in the training data (i.e., the virtual-world data) because the naive model learns to maximize the output probability of words in the vocabulary of the training dataset. Therefore, it cannot deal with answers that contain an entity (e.g., object, place, and person) not included in the virtual world. For example, if "washroom" does not appear in the virtual-world data, the naive model cannot output it as an answer even if "washroom" appears in the real-world events.

In contrast, our QA model learns a general reasoning process for real-world events independent of environments. When sentences about daily stories and a question are given, our model first focuses on sentences that relate to the question. Our model then focuses on the words in the found sentences that might relate to the question, which are used to generate an answer. In the above example, our model first focuses on sentences, including the word "Mary." Because "Where is" in the question specifies the latest location of "Mary," our model focuses on the last sentence and the words in it related to the question. Since "Where" in the question specifies a place, our model focuses on the target word "washroom," which is located after "in." Moreover, our model can output the target word "washroom" even if it did not appear in the training data by coping it from sentences about real-world stories to the answer. This reasoning process can be learned in any environment when sufficient training data are given, and the process can be applied to any real target real environment.

Note that, unlike standard question answering studied in the NLP community, which mainly focused on choosing an answer from multiple choices or selecting an answer range in documents, real-world QA must generate diverse types of answers (e.g., numbers and a sequence of names of entities) by processing a sequence of real-world events. Therefore, our method is designed to be equipped with a module that is responsible for answer generation that efficiently generates an answer by synthesizing information about multiple important events (sentences that require high attention) to facilitate counting the number of occurrences of an event and enumerating the names of the entities related to the question.

As described above, our method focuses on critical sentences and words in given stories. We implement this idea by employing an attention mechanism that extracts important information in neural network inputs. We incorporate event-level attention that computes the weight (importance) of each event (sentence) into our QA model as well as the word-level attention that computes each word's weight. We then generate or copy an answer by employing words with high weights that appeared in the events (sentences) with high weights. By using the word attention distribution of input sequence, this approach output answers that are related to an entity even when it is not found in a virtual-world QA dataset.

C. CONTRIBUTIONS

The following are the contributions of this study:

- We introduced a novel Sim2RealQA framework that uses a QA model that is entirely trained with a virtual-world dataset for solving real-world QA problems.
- To accelerate real-world QA study in the IoT community, we developed real-world and virtual-world QA datasets comprised of daily life sto-

ries collected from an actual house and simulated household environments. The dataset is available: *https://miyatai.org/data/Sim2RealQA.zip*.

- We proposed a new real-world QA model tailored to Sim2RealQA. The proposed model can learn a general reasoning process of real-world QA independent of environments by leveraging an attention mechanism. In addition, the model is designed for real-world QA so that it can generate diverse types of answers about real-world events by synthesizing information about important events detected by the attention mechanism.
- We evaluated our model using the real-world and virtual-world QA datasets. Our experimental results demonstrate that the Sim2RealQA framework with our model accurately solved real-world QA problems without real-world answer labels for training.

In the rest of this paper, we first review studies on sensor-based context recognition methods and question answering and present our method for dataset construction and our real-world QA model tailored to Sim2RealQA. We evaluate it using data collected in real-world and virtual-world environments.

II. RELATED WORK

We solved real-world QA problems with a QA model trained with virtual-world data. To tackle this cross-domain QA, we used a life simulator that generated virtual-world stories for making virtual-world QA datasets. In this section, we introduce sensor-based context recognition, the existing language tasks in a simulated world, simulations to real approaches, and cross-domain QA methods that describe different aspects from ours.

A. SENSOR-BASED CONTEXT RECOGNITION

In the IoT community, sensor-based context recognition methods have been studied, especially context related to human daily activities and indoor positions. Context recognition methods can be roughly grouped into wearable sensing and environment augmentation. The former approach employs such body-worn sensors as accelerometers, microphones, cameras, and Wi-Fi receivers. The latter approach employs sensors embedded in an environment, e.g., cameras, microphones, switch sensors, RFID tags, and Wi-Fi transmitters/receivers. e Activity recognition based on body-worn accelerometers recognizes simple activities such as walking, eating, drinking, and brushing teeth [3], [4], [24]–[26]. Activity recognition based on body-worn cameras recognizes complex activities that involve interactions with objects or other persons, such as eating, talking with someone, and reading [5], [27], [28]. Activity recognition based on object-attached sensors such as switch sensors and RFID tags also categorizes complex activities by sensing interactions with objects [29]–[32]. Several methods also detect a person's activities as well as the objects or those who are interacting with the person of interest.

Indoor positioning methods estimate the indoor coordinates of a signal receiver or a place class, e.g., toilet or kitchen, using wearable sensors [8], [33], [34]. Cameras installed in an environment can also be used for indoor positioning with a person identification technique [35]. Estimated indoor coordinates are usually converted to the name of a room when this information is provided to a user. We assume the above activity recognition and indoor positioning techniques for generating sentences about daily life stories.

B. LANGUAGE TASKS IN A SIMULATED WORLD

For developing intelligent systems that perform language tasks in realistic environments, many studies have used simulators to train models that perform such language tasks as executing navigation instructions with natural language [36]–[39], answering questions about virtual-world situations with embodied agents in a simulated house [40]–[42], and generating the daily household activities of human-like agents [43], [44]. These methods presented in earlier works admirably performed the given tasks because the simulations provide a sufficient amount of labeled data or rewards for training. However, they only solve the language tasks with a simulator. On the other hand, although our framework also uses simulations, its main purpose is to solve real-world language tasks with virtual-world datasets that are comprised of simulation data.

C. SIMULATION TO REAL

Modern machine-learning systems that use deep neural networks require many labeled training datasets to achieve superior performance. To execute a real-world task that often lacks labeled data, transferring machine-learning systems from simulations to the real world has been widely used, such as navigating a robot to find a target object indoors [45], grasping various objects with robotic arms [46], learning to drive from a simulation [47], collision avoidance for drones [48], in-hand manipulation [49], agile locomotion for quadruped robots [50], and the semantic segmentation of actual driving video using a popular video game [51]. The results of these works indicate that simulation plays an important role in training machine-learning systems and increases the real-world task performance. The method proposed and examined in this study also uses the notion of simulations to real data. Unlike earlier works, we specifically leverage simulators to address real-world QA problems.

D. NATURAL ANSWER GENERATION

Question answering aims to automatically answer questions about a given context (e.g., documents, knowledge base, and multimedia data.) There are several ways of answering questions: selecting the span corresponding to the answer from the document [52], [53], choosing one from multiple answer options [19], [54], or generating an answer [20]. For responding to various real-world information needs, the real-world QA task [18] takes the form of generating answers. As with the real-world QA, most existing works use the encoder-decoder framework that takes as input a question and a sequence of words in a given context and then generates answer words [55]–[57]. For augmenting the question and answer pairs to improve QA performance, the methods of generating questions as well as answers have also been proposed [58], [59]. However, these works mainly focus on a single domain for answer generation. In contrast to them, we take a cross-domain QA framework that trains the QA model with source domain data (virtual-world data) and generates answers with target domain data (real-world data.) Another setting of natural answer generation is the setting of multi-turn QA (i.e., dialogue) [60], [61], which generates answer responses based on a given context and a history, including past questions and answers. Our story-based QA can be viewed as a special case of single-turn dialogue, which does not use any past questions and answers. Even though this study's main topic is to generalize the single-turn QA model trained with the virtual-world data to the real-world data, we can naturally extend our story-based QA method to the multi-turn QA method by using the QA history as input.

E. CROSS-DOMAIN QUESTION ANSWERING

For real-world QA tasks, making QA datasets by collecting real-world stories is costly and complicated by privacy issues. We address this difficulty using a cross-domain QA method that uses QA models learned from a source QA dataset to solve target QA problems. Existing cross-domain QA methods are used under supervised, semi-supervised, and unsupervised conditions. Supervised methods use both labeled source and target datasets. They pre-train a QA model using the source dataset and fine-tune it with the target domain [62], [63]. A semi-supervised method uses unlabeled source and labeled target datasets and the unlabeled source dataset to boost the performance of the QA models by adapting a model to the target domain [64], [65]. In contrast to the supervised and semi-supervised QA methods, we use an unsupervised approach based on the difficulty of obtaining real-world, labeled data in the target domain. Under the unsupervised QA scenario, we can use a source domain (virtual-world) dataset for training the models, but target domain (real-world) labels are unavailable. Earlier studies investigated cross-domain QA under unsupervised conditions and demonstrated their usefulness [66]–[69]. However, these works used standard reading-comprehension or QA datasets comprised of Wikipedia entries, web snippets, and newspaper articles. These documents are completely different from the daily living stories used for real-world QA. Moreover, these studies focused on choosing an answer from multiple choices or selecting an answer range in documents. In contrast, real-world QA tasks must generate numbers, "yes/no" answers, and a sequence of words as answers based on the content of a given question for addressing various information needs in real-world situations. For these reasons, we created daily life stories with a life simulator and made a virtual-world QA dataset that resembles a target real-world QA dataset.

III. GENERATING DAILY LIFE STORIES AND CONSTRUCTING DATASET

In this section, we introduce how to create sentences about daily stories from the outputs of activity recognition and indoor positioning systems. After that, we describe the real-world and virtual-world QA datasets constructed in this study.

A. GENERATING STORIES

In this study, we generate sentences related to (i) locomotion and (ii) activity. When an indoor positioning system detects that a person has moved to a room, we generate a sentence of the event based on the following template:

Template for locomotion

[person] moved to the [place.current] from the [place.previous].

Here, [person] is replaced by the name of the individual being tracked. [place.current] is replaced by the name of the current room, and [place.previous] is replaced by a name of the room from which the individual moved. The following is an example sentence generated from this template: "David moved to the toilet from the living room."

As for daily activities, we assume activities that can be performed both by a single person and by multiple persons. When a single-person activity is detected by an activity recognition system, we generate a sentence of the event based on the following template:

Template for single-person activity

[person] [activity] [[activity.object]] in the [place.current].

Here, [person] is replaced by the name of the individual being monitored, and [activity] is replaced by the name of the detected activity. Because [[activity.object]] shows an option, it is replaced by an object's name if the system can detect the object that was used in the activity (e.g., using RFIDs or body-worn cameras). A preposition is inserted before the object name, if necessary (e.g., "with"). Note that in some activities, the object being used can be automatically determined. For example, when toothbrushing is detected, an object must undoubtedly be a toothbrush. [place.current] is replaced by the user's current place detected by the indoor positioning system. The following is an example sentence generated from this template: "Sheldon read a book in the living room."

When a multi-person activity is detected by the activity recognition system, we generate a sentence of the event based on the following template:

Template for multi-person activity

[person] [activity] [[activity.object]] with [activity.person] in the [place.current].

Here, [activity.person] is replaced by the name(s) of a person detected as a member of the activity (e.g., using body-worn cameras or proximity sensors). The following is an example sentence generated from this template: "Sheldon talked with David in the living room." We assume that the activity recognition and indoor positioning systems generate sentences based on the above templates.

B. DATASET CONSTRUCTION

We assume the above context recognition systems and introduce the construction of real-world and virtual-world QA datasets.

1) Story Collection Method

We generated story events for our datasets by observing real/virtual environments depicted in Figs. 2 and 3.

Real-world stories: We attached a wearable camera to each subject to obtain stories in the real house environment depicted in Fig. 2. Annotators watches the captured videos and manually created sentences based on templates described in Section III-A. We manually generated story sentences because this study's purpose is to construct precise QA datasets that are independent of sensor systems as well as to validate the effectiveness of our proposed Sim2RealQA framework.

In the experiment, we generally followed an earlier study of real-world QA tasks [18]. Real-world stories consist of sentences that describe various daily living activities in residential settings. To collect more diverse daily activities than in a laboratory setting, we used a semi-naturalistic collection protocol [1]. We attached a Tobii Pro wearable eye tracker to five subjects who repeatedly performed 20 daily activities ten times in six different places: bathroom, bedroom, entrance, kitchen, living room, and washroom. For example, the subject makes coffee in the kitchen, takes it to the living room, and drinks it while watching TV. During the data collection, we captured first-person videos of their daily activities and obtained ten real-world stories per person: 50 stories. Two annotators labeled these first-person videos using a sequence of sentences that described what they are doing, when, and where inside a house. Fig. 4 (top) shows examples of the annotated real-world stories. We obtained 7,369 story events (697 unique) about their daily activities. Each story has 147 $\pm$ 8 sentences and 1,338 $\pm$ 76 words on average.

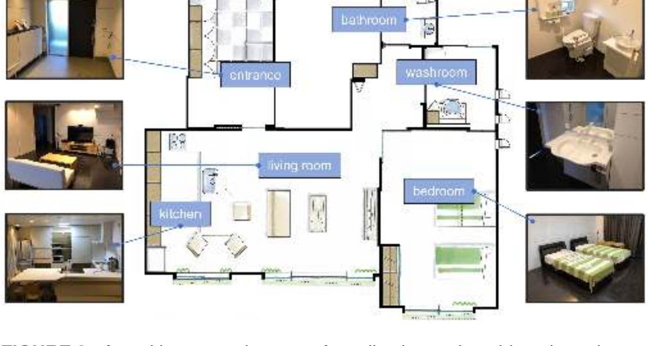


FIGURE 2. Actual house environment for collecting real-world stories, where a person performs a variety of daily life activities at each location

Virtual-world stories: For collecting virtual-world stories, we used a life-simulation game called The Sims[2], which replicates the life of an individual person in a virtual world. In contrast to recent simulators that imitate household environments [70]–[73], The Sims simulator easily and automatically generates many human-life stories about virtual-world residents called Sims. In their respective environments, they make their own life choices based on the available electrical appliances and furniture in their house and to match their physical and mental needs (e.g., hunger, companionship, hygiene, entertainment, health, and bodily functions), which are represented by their interior parameters. For example, when a Sim's hunger parameter decreases, she removes food from her refrigerator, moves to the dining room table, and starts eating. We can easily customize the room layout and include such furniture and appliances as beds, sofas, coffee-makers, and PCs in the house environment where the Sims live. With these advantages of a life simulator, we obtain more realistic and diverse daily living stories than fictional stories [21], [54], [74], books [19], [20], and movie scripts [75] where detailed human activity logs are not recorded. Note that this work used The Sims due to its simplicity, but other life simulators (e.g., VirtualHome [43]) can also be used.

We simulated daily activities by preparing three shared housing environments with typical households using The Sims because the real target environments and residents are actually unknown. Each environment has a kitchen as well as a dining room, a living room, a bathroom, and bedrooms with appropriate objects for daily life. Fig. 3 depicts three environments used for the data collection. Using these house settings, we simulated the daily lives of ten family units comprised of 16 adults. With the life-simulation game, we collected 30 days of daily activities per family unit (i.e., 300 stories). Two annotators manually created story events by watching the recorded game-play videos. We obtained 54,770 story events (7,218 unique) about the daily activity events. Each story has 183 $\pm$ 58 sentences and 1,568 $\pm$ 523 words on average. Fig. 4 (bottom) shows examples of the annotated virtual-world stories. Table 1 show a list of names of entities used for creating sentences about daily life stories in the real and virtual worlds.

2) QA Creation Method

For each story, we made question and answer pairs to construct QA datasets, i.e., question, answer, and story triplets. We made a template of 22 QA tasks related to world situations following previous real-world QA work [18]. Table 2 shows the QA template with which we generated questions about each task. First, we randomly select the positions where questions are inserted in each story for each person in both worlds and then generate a question using a question template and events before the question's position. We repeat this process until generating 20 questions per task. Then, we generate answers to a given story by an oracle QA which can

[2]https://www.ea.com/games/the-sims

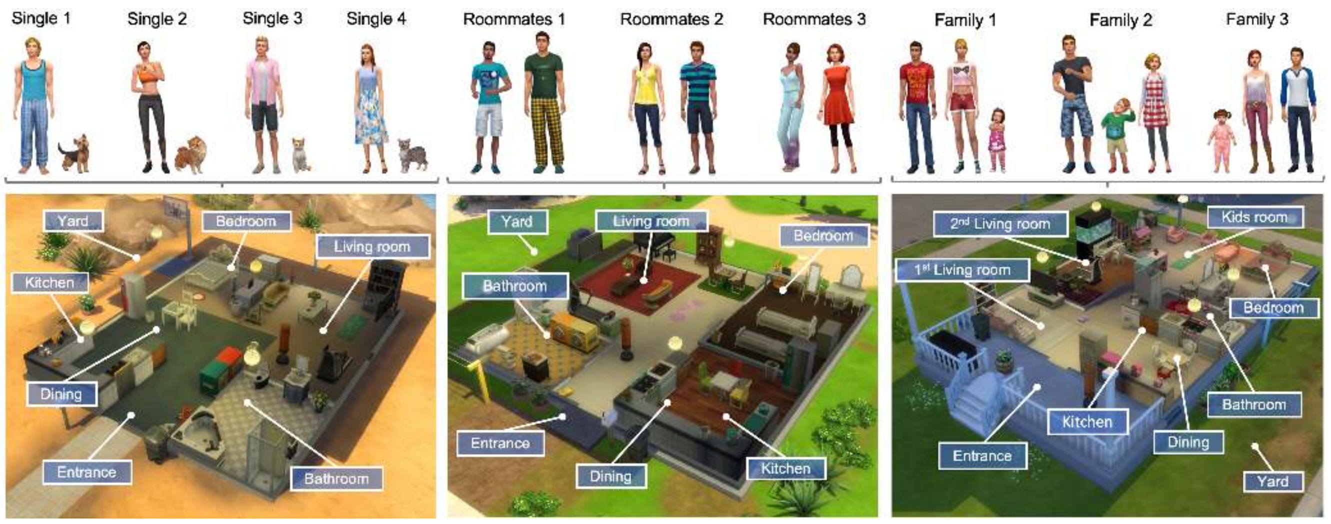


FIGURE 3. House environments in The Sims 4 for simulating daily life: Environment 1 (**left**) simulates a single-person household. Environment 2 (**center**) simulates a shared house. Environment 3 (**right**) simulates a nuclear family. We use artificial personages as described above for each environment to obtain daily life stories.

TABLE 1. List of names of entities used for generating daily life stories

	Real world	Virtual world
Subject	Courteney, David, Jennifer, Lisa, Matt	Amy, Bernadette, Chandler, Dharma, Gregory, Howard, Joey, Leonard, Monica, Penny, Phoebe, Rachel, Ross, Sheldon, Skyler, Walter
Predicate	brush, close, drink, eat, flush, hold, look at, make, moisten, mop, move, open, pick up, pour, pour off, put, put on, read, remove, scoop, sit down on, spread, stand up from, stir, take out, throw away, turn off, turn on, unroll, wash, watch, wipe	add, bring in, brush, call from, change, change into, clean, clean on, close, cut, do pushups, do yoga, dress, drink, eat, feed, fix, flush, fry, get off, get up from, grill, hang out, hit punching, hold, hold cutting, hug, kiss, knead, listen to, look at, mop, move, open, pat, pet, play, play with, pour, pour off, pull, put, put away, put on, read, replenish, run on, saute, season, set, shoot, sit down on, sleep in, sleep on, stand up from, stew, stir, sweep, take, take off, take out, talk with, taste, throw away, turn off, turn on, undress, use, wash, watch, water, wipe
Object	air conditioner, bed, book, butter knife, coffee, coffee bottle, coffee powder, cracker, cracker pack, cracker sand, crackers, cream cheese, cream cheese pack, cup, cupboard, dishwasher, electric kettle, electric kettle lid, face, floor, hands, hot water, light, plate, refrigerator, shoes, slippers, sofa, spoon, table, teeth, television, toilet, toilet lid, toilet paper, toothbrush, water, wristwatch	BBQ grill, auto feeder, bag, ball, basketball, bath, bed, beef, board, bowl, bread, casual wear, cat, chair, channel, chicken, child, clothes, coffee, coffee beans, coffee maker, comic book, computer, cup, desk, dining table, dirt, dishwasher, dog, doll house, dressing table, dryer, egg, fertilizer, fish, floor, garbage box, glass, guitar, hands, hot water, ice cream, ice cream maker, kitchen counter, kitchen sink, large plate, lavatory sink, magazine, mail, mail box, microwave, milk, mushroom, music, night clothes, noodle, novel, oven, photo book, piano, picture book, plant, plate, pork, potato, recipe book, refrigerator, rice, rumba, seafood, shower, situps, smartphone, sofa, speaker, sportswear, sugar, tea, tea leaf, tea maker, television, textbook, tofu, toilet, trash, trash bag, treadmill, tropical fishes, vegetable, violin, washer, water, weeds
Location	bathroom, bedroom, entrance, kitchen, living room, washroom	1st living room, 2nd living room, bathroom, bedroom, dining, entrance, kids room, kitchen, living room, yard

accurately answer all the questions about both real-world and virtual-world QAs using the syntax structure of the questions and stories. For example, given a story "Tom washed the plate in the kitchen. Tom moved to the living room from the kitchen.", we insert a question template "Where is [person]?" after the first event. Then, we automatically fill in the question template according to the first event's content and generate a question, "Where is Tom?". Finally, we create an answer "kitchen" by the oracle QA. Also, when a second question is inserted after the second event, the same question "Where is Tom?" and a different answer "living room" will be generated. An oracle for generating answers was also used in an earlier study [18], [21], [76], [77]. Note that we only use the oracle QA to validate the concept of Sim2RealQA, which is not available in the actual case. Due to the diversity of natural language questions, the oracle QA is not practical in real-world situations compared to learning-based approaches which can learn such diversity from data. "No answer" tokens are used if a question has no answer. We used the same data format for all the tasks as the bAbI dataset [21]. We also show examples of real-world and virtual-world QAs in Fig. 4. Their activities are identical in many cases, but the persons, objects, places, and daily life patterns in both worlds are sometimes different. In particular, QA models must output answers to the unknown entities that appear in the real world, but not in the virtual world (e.g., "Lisa" and "washroom," in Fig. 4) and generate such answers as numbers, "yes/no," and several entities (e.g., "entrance, kitchen, living room," in Fig. 4) in response to various information needs in the real world. In fact, 28% of the answer words in the target domain

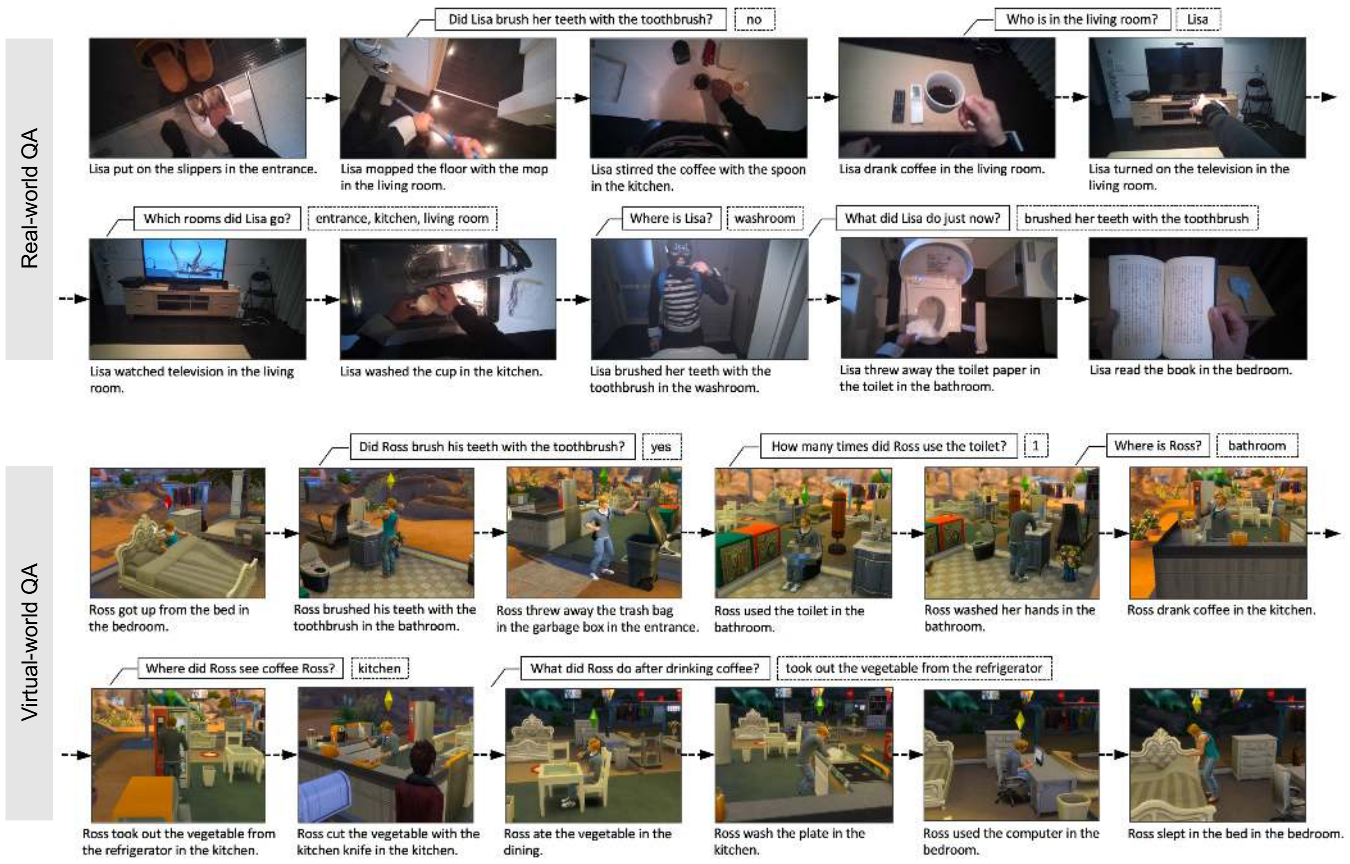

Real-world QA
Did Lisa brush her teeth with the toothbrush?
no
Who is in the living room?
Lisa
Lisa put on the slippers in the entrance.
Lisa mopped the floor with the mop in the living room.
Lisa stirred the coffee with the spoon in the kitchen.
Lisa drank coffee in the living room.
Lisa turned on the television in the living room.
Which rooms did Lisa go?
entrance, kitchen, living room
Where is Lisa?
washroom
What did Lisa do just now?
brushed her teeth with the toothbrush
Lisa watched television in the living room.
Lisa washed the cup in the kitchen.
Lisa brushed her teeth with the toothbrush in the washroom.
Lisa threw away the toilet paper in the toilet in the bathroom.
Lisa read the book in the bedroom.
Virtual-world QA
Did Ross brush his teeth with the toothbrush?
yes
How many times did Ross use the toilet?
1
Where is Ross?
bathroom
Ross got up from the bed in the bedroom.
Ross brushed his teeth with the toothbrush in the bathroom.
Ross threw away the trash bag in the garbage box in the entrance.
Ross used the toilet in the bathroom.
Ross washed her hands in the bathroom.
Ross drank coffee in the kitchen.
Where did Ross see coffee Ross?
kitchen
What did Ross do after drinking coffee?
took out the vegetable from the refrigerator
Ross took out the vegetable from the refrigerator in the kitchen.
Ross cut the vegetable with the kitchen knife in the kitchen.
Ross ate the vegetable in the dining.
Ross wash the plate in the kitchen.
Ross used the computer in the bedroom.
Ross slept in the bed in the bedroom.

FIGURE 4. Examples of real-world and virtual-world QA datasets using daily life stories collected in an actual house and a life simulator. Our QA datasets ask questions about daily life stories in both worlds.

TABLE 2. Question and answer template on each QA task, where words [person], [activity], [object], and [place] are randomly extracted from given stories of QA datasets. '+' indicates multiple entities.

Task	Question	Answer
1	Is [person] in [place]?	yes/no
2	Did [person] [activity] just now?	yes/no
3	Was [person] in [place]?	yes/no
4	Did [person] [activity] ?	yes/no
5	Where is [person]?	[place]
6	Where did [person] see [object]?	[place]
7	Where was [person] before [place]?	[place]
8	Where was [person] after [place]?	[place]
9	Where did [person] see [object] before [place]?	[place]
10	Where did [person] see [object] after [place]?	[place]
11	Who is in [place]?	[person]+
12	Who was in [place]?	[person]+
13	Who [activity]?	[person]+
14	Where did [person] [activity]?	[place]
15	What did [person] [activity]?	[object]
16	Which rooms did [person] go?	[place]+
17	What did [person] do just now?	[activity]
18	What did [person] do before [activity]?	[activity]
19	What did [person] do after [activity]?	[activity]
20	How many rooms did [person] go?	[number]
21	How many times did [person] [activity]?	[number]
22	How many times did [person] see [object]?	[number]

(real world) do not appear in the source domain (virtual world). To exploit the simulation data and further improve the real-world QA performance, we need to address these differences between the real and virtual worlds.

IV. REAL-WORLD QA MODEL FOR SIM2REALQA

A. OVERVIEW

We introduce our real-world QA model for the Sim2RealQA framework trained on the source QA examples (i.e., question, answer, and story triplets) of the virtual world, which can output correct answer when the target story and question sets of the real world are given. To achieve this generalization of real-world QA problems, we addressed the unknown entities that do not appear in the source QA examples, but which do appear in the target QA examples that simultaneously capture the relations over multiple events related to a given question.

The overall architecture of our QA model is shown in Fig. 5. The model mainly consists of five layers: (i) embedding, (ii) context, (iii) attention , (iv) matching, and (v) answer. The layers (i-iv) are parts of a dynamic memory module inspired by a dynamic memory network [78], [79]. Layer (v) is part of the pointer generator module inspired by pointer generator networks [80]. For a brief explanation, we consider a sentence in a story an *event* and a sequence of sentences about a daily story *story events*. In our model, the input consists of events (a daily-life story) and a question. First, the embedding layer extracts their word feature vectors (i.e., word embeddings). Second, the context layer takes the word embeddings as input, computes the sequential dependencies of the words in the question and each event, and outputs a

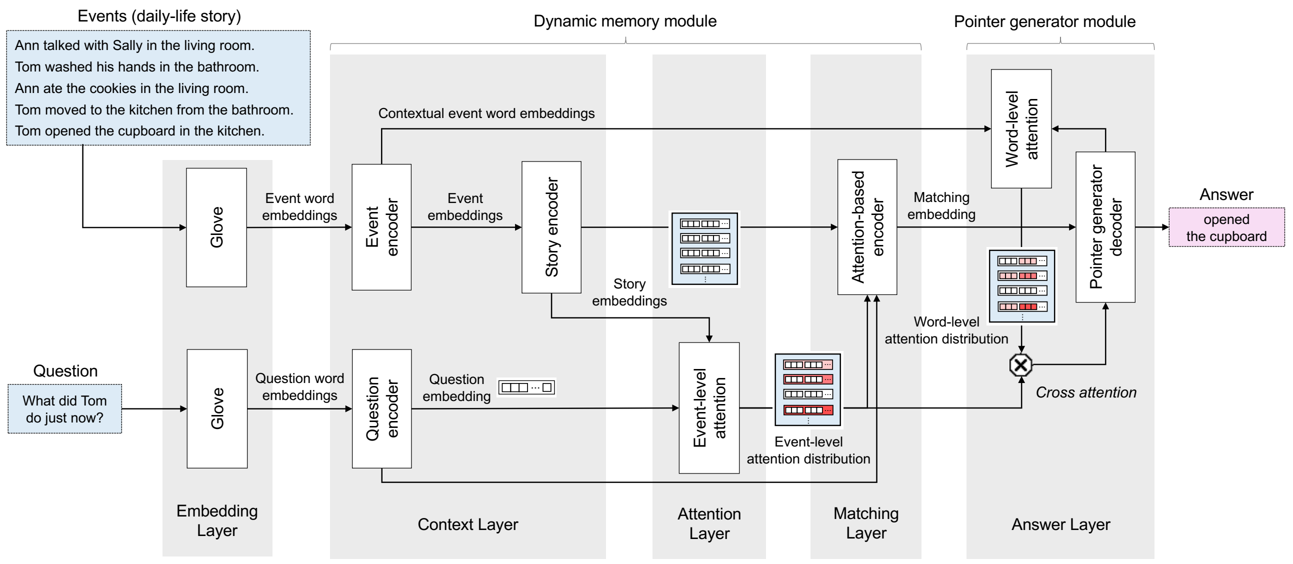


FIGURE 5. Overall model architecture for solving real-world QA problems with Sim2RealQA framework. Given sentences of a daily life story and a question, the QA model outputs a corresponding answer sentence through inference by five layers.

question embedding and event embeddings. In addition, this layer takes event embeddings as input and calculates the story embeddings that capture the context of story events. The attention layer takes the question and story embeddings as input and computes the event-level attention that represents an event's importance for a given question. The matching layer aggregates the events weighted with event-level attentions and outputs the matching embedding. Because the matching embedding is a vector that represents the association between a question and its relevant events, it is used for decoding answers. Finally, the answer layer outputs an answer sentence based on the two types of word weights from the vocabulary and word attention distributions. The vocabulary distribution, which is a probability distribution over all the words in the vocabulary of the training dataset, is calculated based on the hidden state of an RNN language model trained for predicting the answer words. By using this distribution, the model can generate answer words from the fixed vocabulary used in training. The word attention distribution, which is a probability distribution over the words in the input sequence, is calculated based on the cumulative attentions of the input words for generating the answer words. By sampling words from this distribution, the model can output unseen entities as an answer when such entities are included in the input story. We extended this word attention distribution with event-level attention that represents the relevance between a question and events because relevant events to a question are likely to contain words suitable for the answer. Finally, the answer decoder in the layer recurrently generates a sequence of answer words by integrating the vocabulary and the extended word attention distribution. We explain each component of our model in detail.

B. EMBEDDING LAYER

The model's input is the question and story events. First, we convert each word in the question and story events into vectors that represent the semantics of words. For the vector representation of the word in the events and the question, we use the Glove model trained with Gigaword5 + Wikipedia2014 corpus [81]. This layer's outputs are question word embeddings $Q_w \in \mathbb{R}^{n \times d_w}$ and event word embeddings $E_w \in \mathbb{R}^{m \times d_w}$, where n is the length of a question (i.e., the number of words in it), m is the length of an event (i.e., the number of words in it), and d_w is the size of the word embedding. The event word embedding is created by each event in a story.

C. CONTEXT LAYER

This layer models a sequence of words (i.e., question and event) and a sequence of events (i.e., story) using the question, event, and story encoders. To model the long-term dependencies in the input sequence, we encode the sequences of the words and events using a bidirectional-GRU (Bi-GRU) [82], [83], which is a special kind of recurrent neural network. The question encoder outputs the GRU's hidden state after reading question word embeddings Q_w as question embedding $q \in \mathbb{R}^{d_h}$, where d_h is the size of the hidden state of Bi-GRU. The question embedding corresponds to modeling a sequence of words in the question. The event encoder reads event word embeddings E_w and outputs GRU's hidden state as the event embedding. This resembles the modeling of a sequence of words in the event. The event embeddings of all events are denoted as $E \in \mathbb{R}^{l \times d_h}$, where l is the length of the story (i.e., the number of events in it). In addition, we use the GRU outputs as the contextual event word embeddings $E_c \in \mathbb{R}^{m \times d_h}$ when reading the event word embeddings. The contextual event word embeddings model the sequential word representations in each event to compute the word-level attention distribution for answer decoding. We also use the GRU outputs (i.e., the hidden states of the story encoder) when reading the event embeddings as story embeddings. The story embeddings of all the events are denoted as $S \in \mathbb{R}^{l \times d_h}$, which resembles modeling of the context of events in a story.

D. ATTENTION LAYER

This layer computes how essential an event is to a given question using event-level attention. It is a useful clue to find the relevant events to a user's question in a long story. In addition, this information helps enumerating name of entities a particular event related to a question. Using story embeddings S and question embedding q, the event-level attention computes the weight of each event. For example, this attention increases the weight of events (e.g.,"Tom washed his hands in the bathroom," "Tom moved to the kitchen from the bathroom," and "Tom opened the cupboard in the kitchen" in Fig. 5), including the words in a question (e.g., "Tom"). The attention value of k^{th} event is given by

$$\beta_k = \mathrm{softmax}(W_\beta \tanh(W_z z_k + b_z) + b_\beta), \qquad (1)$$

where W_β, W_z, b_β, and b_z are the learnable parameters, $z_k = [s_k \circ q; |s_k - q|]$, and $s_k \in S$ is the story embedding of the k^{th} event. Here ; denotes horizontal vector concatenation, $\circ$ is an adamal product, and $|\cdot|$ is the absolute value for each element. Because $s_k \circ q$ represents the similarity between two vectors and $|s_k - q|$ represents their distance, z_k represents the relationship between the story event and a question. We use this event-level attention distribution β for weighting story events by their importance to a question in the next matching layer.

E. MATCHING LAYER

This layer encodes the story and question embedding as a question-story matching embedding (matching embedding for short) with an attention-based encoder. First, the attention-based encoder sequentially weights k^{th} story embedding $s_k \in S$ with corresponding event-level attention value β_k and aggregates them by

$$c_k = (1 - \beta_k) \circ c_{k-1} + \beta_k \circ s_k. \qquad (2)$$

The output of encoder c after reading all story embeddings S is the weighted sum of the story embeddings by relevance to the questions, which emphasize recent events. To predict a question's answer based on question embedding q and final weighted story embedding c, this layer computes matching embedding m:

$$m = \mathrm{ReLU}(W_m[c; q] + b_m), \qquad (3)$$

where W_m and b_m are the learnable parameters. Here ReLU is a Rectified Linear Unit [84]. The matching embedding holds the relevance information between a given question and story events computed by the event-level attention which gives a large weight to events related to the question. Therefore, it contains important clues to answer the question. Then, we use matching embedding m as the initial hidden state of RNN-based decoder h_0 in the answer layer for bringing the matching results between a question and a story to answer decoding.

F. ANSWER LAYER

This layer in the pointer-generator module generates a sequence of answer words based on the matching results between the question and the story using the pointer-generator decoder. Fig. 6 shows its overview. First, the RNN-based decoder with a hidden layer initialized by matching embedding reads a $\langle START \rangle$ token that indicates the beginning of the answer sentence and updates the internal state. Based on this updated hidden state, the decoder then generates the next word based on the (i) vocabulary distribution and (ii) the word attention distribution. The decoder computes the vocabulary distribution to predict the next answer word in the vocabulary used in training. It also computes the word attention distribution to address the unknown entities that do not appear in the training dataset. The decoder uses the word attention distribution to copy words from the input story events with a pointing mechanism [80], [85]–[87]. Second, this layer updates this word attention distribution using the event attention distribution from the encoder and computes the extended word attention distribution to actively reflect a question's intention about an answer. Finally, the decoder samples the word with the highest probability based on the final distribution that combines the extended word attention distribution and the vocabulary distribution and uses it for the next decoder input. By repeating this process, the answer layer can output multiple types of answer sentences, e.g., word sequences, word sets, as well as a single word. In this section, we first introduce the conventional pointer-generator decoder used as a part of the answer layer and explain its extension that addresses unknown entities by considering the question's intention.

1) Pointer-generator decoder

To predict a sequence of answer words, the pointer-generator decoder sequentially generates words by using previously generated words as additional input at each decoding step. At each decoding step, the decoder outputs answer word a with word-level attention that tells the decoder which words in the story are useful for predicting the next word. The word-level attention distribution α^t at each decoding step t is defined:

$$o_j^t = v^T \tanh(W_e e_j + W_h h_t + b_e) \qquad (4)$$
$$\alpha^t = \mathrm{softmax}(o^t), \qquad (5)$$

where e_j denotes the j^{th} contextual event word embedding from the context layer, h_t denotes a t^{th} decoder hidden state, and v, W_e, W_h, and b_e are learnable parameters. Therefore, the word-level attention is computed based on the word-level encoder's hidden state (i.e., contextual event word embedding) and the decoder's hidden state. By using this word-level

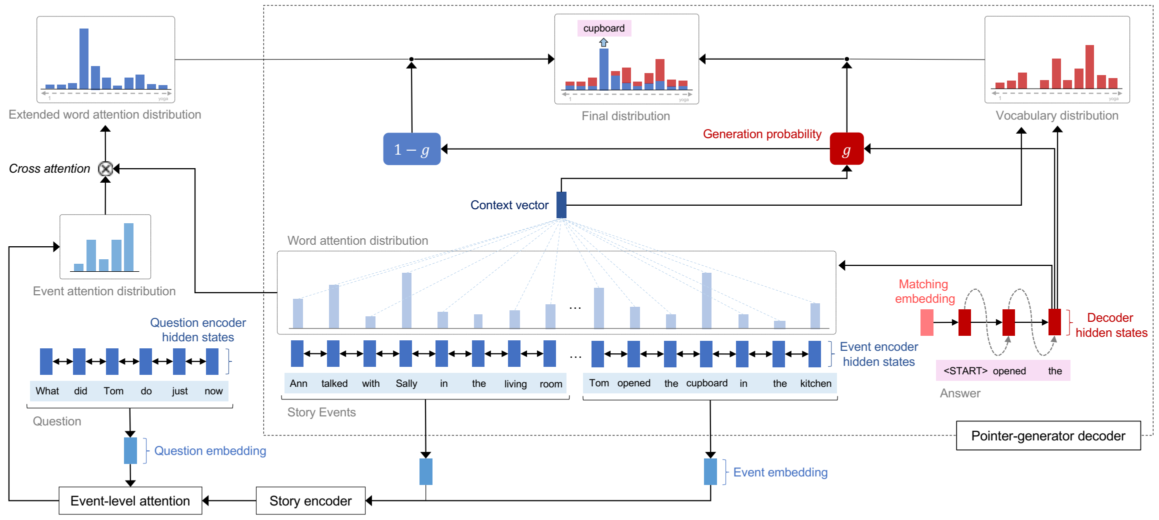


FIGURE 6. Overview of answer layer: Point-generator decoder sequentially generate answer words or copy them from the input in autoregressive manner by integrating vocabulary and extended word attention distribution. The extended word attention distribution holds information about question's intention by updating the original word attention distribution with the event attention distribution.

attention, the decoder produces answer words based on the vocabulary distribution:

$$P_{vocab} = \mathrm{softmax}(W_1(W_2[h_t; u_t] + b_2) + b_1), \qquad (6)$$

where W_1, W_2, b_1, and b_2 are learnable parameters. Here $u_t = \sum_j \alpha_j^t e_j$ is called a *context vector* that represents a sequence of words relevant to the current decoding step. Therefore, the vocabulary distribution is computed based on the decoder's hidden state and the word-level encoder's hidden states weighted by word-level attention that indicates which input words are important for the current decoding step. By using this distribution, the decoder can create any words from a fixed vocabulary. However, the word probability from P_{vocab} with regard to unknown entities almost becomes zero because they did not appear during the training phase.

To address this problem, the pointer-generator decoder uses words in the story as outputs by using the following cumulative word attention distribution:

$$P_{copy} = \sum_{j:w_j=w} \alpha_j^t, \qquad (7)$$

where w_j is the j^{th} word in the input story. Copy distribution P_{copy} assigns probability to the words of the unknown entity using the attention values of the input words. For example, the cumulative attention value of "cupboard" becomes 0.2 when it appears in the input story and its word-level attention value is 0.2. Due to this trick, the decoder can output "cupboard" as the answer, even if "cupboard" does not appear in the training dataset, i.e., the virtual world.

Finally, the decoder outputs next answer word a based on the final distribution:

$$P_{final}(a) = gP_{vocab}(a) + (1-g)P_{copy}(a), \qquad (8)$$

where g is a *generation probability* $g \in [0, 1]$ that decides whether the decoder generates a word from the vocabulary from P_{vocab} or uses a word from word attention distribution P_{copy}. Generation probability g is given by

$$g = \sigma(w_h^T h_t + w_u^T u_t + w_a^T a_t + b_g), \qquad (9)$$

where vectors w_u, w_s, w_a, and b_g are learnable parameters, $a_t \in \mathbb{R}^{d_w}$ is a word embedding of a generated answer word at the previous step $t-1$, and σ is the sigmoid function. Therefore, g is computed based on the input context, the current decoding state, and the decoder input. Distribution P_{final} produces unknown entities based on the word attention distribution of source P_{copy}. We assume that this pointer-generator mechanism is helpful for producing answers, including the unknown entities caused by the differences between the real and virtual worlds.

Unfortunately, this pointer-generator mechanism only uses word-level attention for copying words; it ignores the relations between story events and a question, i.e., event-level attention. We assume the encoder's event-level attention indicates the important events in which users are interested. Since the words in these events are worth copying, we integrate the word-level and event-level attentions for decoding.

2) Cross attention

To identify suitable words for copying, we extend word-level attention α for using input words in the events (Eq. 7) using event-level attention β (Eq. 1). Extended word-level attention α'_j of the j^{th} word in the input text is defined:

$$\alpha'_j = \frac{\alpha_j \times \beta_{[j]}}{\sum_i \alpha_i \times \beta_{[i]}}, \qquad (10)$$

where $[j]$ is the index of the story event that contains the j^{th} word. The definition holds. When both word-level attention α_j and event-level attention $\beta_{[j]}$ are high, new α'_j

becomes high. Thereby, we can incorporate the user intention described by a question sentence in the pointer-generator decoder. We use this extended word-level attention distribution α' for P_{copy} instead of the original one, α, in Eq. 7.

V. EXPERIMENTS

We evaluated our Sim2RealQA framework using real- and virtual-world QA datasets. First, we investigated how well our model performed in a Sim2RealQA setting and ascertained what components of QA models contribute to Sim2RealQA. Then we investigated the capability of Sim2RealQA with our model and baselines.

A. EXPERIMENTAL SETUP

1) Methods

We conducted our empirical investigation using the following models, each of which has different modules. By comparing them, we can ascertain what components determine the generalizations when using Sim2RealQA:

- **RNN** is a standard neural QA model based on sequence-to-sequence (Seq2Seq) [88] that encodes the story and a question and then decodes answer words.
- **RNN-AT** is the Seq2Seq-based QA model, which uses a word-level attention mechanism [89], [90]. This mechanism considers the input context for predicting answer words at each decoding step in addition to the above RNN method.
- **RNN-PG** uses the pointer-generator decoder [80] for predicting answers in addition to RNN-AT. The method can use input word distribution using word-level attention to generate or copy words from inputs in the decoding phase.
- **DMN** uses the dynamic memory module for encoding a story and a question relation with their mutual relevance using event-level attentions, which is a special case of the dynamic memory network [78], [79]. The model uses the same decoder as RNN.
- **DMN-PG** uses the pointer-generator decoder in addition to the above DMN. Moreover, this method uses the relevance between story events and a question with the event-level attention distribution to find useful words for copying in the important event for a question.

DMN-PG is our proposed model. We assumed that the pointer-generator decoder and the relevance information between story events and questions further improves the Sim2RealQA performance by addressing the gap between the virtual and real worlds.

In addition to these baselines, we prepared a frequent answer baseline and question-only baselines to check the biases of the real-world and virtual-world QA datasets because question-only methods are competitive in some QA tasks [91], [92]. We prepared the following baselines:

- **Q-Prior** uses the most popular answer per task described in Table 2. We used the frequent answers in the source domain (virtual-world) for predicting answers in the test phase.
- **RNN (Q)**, **RNN-AT (Q)**, and **RNN-PG (Q)** are almost identical to the RNN, RNN-AT, and RNN-PG methods, but they use questions only for decoding answer words.

2) Parameter settings

We trained all the methods with Adam [93] using a learning rate of 0.0001 and a batch size of 20 until 32 epochs were reached. We used early stopping if the accuracy of the validation split in the source domain did not increase for 10 epochs. A null symbol was used to pad them all to a fixed size. We did not update the word vectors during training. The embedding of the null symbol was constrained to zero. For all the RNN encoders and decoders, we used a GRU [82] with a single hidden layer. For the RNN encoders, we used a bidirectional GRU. For all the methods, we selected the following dimensions: 128 word embeddings and 256 hidden states. We set a dropout [94] value of 0.5.

3) Evaluation settings

To assess our proposed framework, we compared the prepared methods with two real-world and virtual-world QA datasets. For training, we divided the virtual-world QA dataset into 169K, 21.1K, and 21.1K examples for the training/valid/test data. For evaluation, we divided the real-world QA dataset into 17.6K, 2.2K, and 2.2K examples for the training/valid/test data. For Sim2RealQA, we trained all the models using the virtual-world training data and evaluated them with the test data from the real world. In this case, the labels in the target domain (real-world) were withheld. We evaluated all the methods under the ParlAI framework [95] with an accuracy measure as an evaluation metrics that computed the exact matches between the predicted answer words and the ground truth. This report describes the QA performances by averaging the results of five training runs based on different initialization values.

B. RESULTS AND ANALYSIS

In this section, we evaluated the performance of the prepared methods over two worlds and explored the generalization of the models trained on a virtual-world dataset to a target real-world dataset. Models learned with virtual-world data, which hold high generalization ability, need to show high performance even when being tested with real-world data.

1) What factors fuel generalization in the real world?

First, we investigated how well our model performed in a Sim2RealQA setting and ascertained what components improved Sim2RealQA's performance. In Table 3 (bottom), we show the Sim2RealQA performance of the prepared methods over all the QA tasks on average. The learning-based approaches (RNN, RNN-AT, RNN-PG, DMN, and DMN-PG) significantly outperformed a simple frequency-based approach, Q-Prior, which uses the most popular answers in the source domain for each task. Moreover, these methods using both stories and questions for QA significantly outperformed RNN (Q), RNN-AT (Q), and RNN-PG (Q), all of which only

TABLE 3. Accuracies of methods over 22 QA tasks. The best result per row is marked in bold typeface.

Task	Q-Prior	RNN (Q)	RNN-AT (Q)	RNN-PG (Q)	RNN	RNN-AT	RNN-PG	DMN	DMN-PG
1	0.510	0.648	0.626	0.692	0.860	**0.938**	0.916	0.908	**0.938**
2	0.430	0.576	0.570	0.568	0.642	0.684	0.782	0.628	**0.834**
3	0.730	0.590	0.590	0.564	0.554	0.784	0.896	0.888	**0.934**
4	0.540	0.534	0.516	0.534	0.488	0.532	0.596	0.534	**0.606**
5	0.600	0.248	0.402	0.430	0.780	0.854	0.996	0.880	**1.000**
6	0.560	0.610	0.562	0.588	0.638	0.716	0.742	0.668	**0.796**
7	0.150	0.146	0.158	0.172	0.240	0.724	0.850	0.370	**0.880**
8	0.200	0.144	0.116	0.158	0.236	0.764	0.752	0.496	**0.920**
9	0.000	0.328	0.302	0.282	0.366	0.402	0.492	0.398	**0.520**
10	0.000	0.454	0.424	0.404	0.522	0.558	0.540	0.560	**0.578**
11	0.000	0.000	0.000	0.000	0.000	0.000	0.996	0.000	**1.000**
12	0.000	0.000	0.000	0.000	0.000	0.000	0.994	0.000	**1.000**
13	0.000	0.000	0.000	0.000	0.000	0.000	**1.000**	0.000	**1.000**
14	0.610	0.640	0.628	0.604	0.666	0.718	0.730	0.648	**0.722**
15	0.040	0.140	0.130	0.136	0.148	0.150	0.270	0.148	**0.364**
16	0.000	0.000	0.000	0.000	0.016	0.030	0.142	0.006	**0.264**
17	0.000	0.000	0.002	0.002	0.020	0.238	**0.810**	0.026	0.754
18	0.000	0.042	0.046	0.046	0.036	0.056	0.116	0.040	**0.164**
19	0.000	0.014	0.016	0.014	0.024	0.032	0.088	0.018	**0.220**
20	0.360	0.360	0.290	0.360	**0.370**	0.352	0.236	0.312	0.320
21	**0.850**	0.824	0.816	0.830	0.842	0.838	0.842	0.820	0.838
22	0.410	0.276	0.314	0.306	0.414	0.432	**0.444**	0.318	0.364
all	0.272	0.299	0.296	0.304	0.357	0.446	0.647	0.394	**0.683**

use questions. These results indicate that training on a virtual-world QA dataset for reasoning over story events in response to a question effectively and accurately solves real-world QA problems. In addition, RNN-PG and DMN-PG significantly outperformed RNN, RNN-AT, and DMN, which do not use the pointer-generator decoder, suggesting that the pointer-generator mechanism further improves the Sim2RealQA performance. Note that our DMN-PG outperformed the other methods, indicating the benefit of integrating the word- and event-level attentions for the pointer-generator mechanism.

We also compared our model's QA performance for each task to the others in Table 3. Across most tasks, RNN-PG and DMN-PG significantly outperformed the others. In particular, RNN-PG and DMN-PG achieved success in tasks 11, 12, and 13, which require answering questions about the names of people; Q-Prior, RNN (Q), RNN-AT (Q), and RNN-PG (Q), RNN, RNN-AT, and DMN failed because no real-world people appeared in the virtual world (i.e., training dataset). The models with the pointer-generator decoder extracted such unknown persons from the given real-world story events and produced them as answers, but the others could not. For example, task 11's results in Fig. 7 (center left) show that DMN-PG and RNN-PG predicted the correct answer "David" who only appears in the real world, but RNN could not. For task 13 in Fig. 7 (center right), RNN also predicted incorrect answer "Howard" who only appears in the virtual world. In addition, the models without the pointer-generator decoder failed to predict unknown places that do not appear in the virtual world. Both RNN-PG and DMN-PG output them as an answer. For example, task 5's results in Fig. 7 (top left) show that DMN-PG and RNN-PG predicted the correct answer "washroom", which only exists in the real world; RNN incorrectly output "bathroom". Both RNN-PG and DMN-PG significantly outperformed Q-Prior, RNN (Q), RNN-AT (Q), RNN-PG (Q), RNN, RNN-AT, and DMN in task 17, which requires answers about activities because they are often different between worlds. The models without the pointer-generator decoder failed to provide answers. For example, task 17's results in Fig. 7 (bottom right) show that DMN-PG and RNN-PG predicted the correct answer, "poured hot water into the cup", which only took place in the real world, and RNN incorrectly output "close to the ice cream maker" that took place only in the virtual world. These results posit compelling evidence that the pointer-generator mechanism is necessary for handling unknown entities caused by the gap that separates the virtual and real worlds. Moreover, the proposed DMN-PG outperformed RNN-PG in difficult tasks 7, 8, 16, 18, and 19, where the models have to answer by referring to multiple events in a long story. DMN-PG has the ability to find multiple relevant events to a question for predicting answers in contrast to RNN-PG. For example, task 8's results in Fig. 7 (top right), which require understanding the context of the place, show that DMN-PG predicted the correct answer, but not RNN-PG. That is because it does not have an event-level attention mechanism that helps it find important clues about the correct answer in a long story. For task 16, which requires multiple events to answer questions, DMN-PG predicted the correct answer, "bedroom, entrance, kitchen, living room", while RNN-PG predicted an incorrect answer: "bedroom, kitchen, living room". This is because the word-level attention in the RNN-PG is difficult to find and memorize answer words,

Task 5: Where is [person]?

Story: · · ·
Lisa sat down on the bed in the bedroom.
Lisa wiped the table with the duster in the bedroom.
Lisa read the book in the bedroom.
Lisa stood up from the bed in the bedroom.
Lisa looked at the wristwatch in the bedroom.
Lisa turned off the light in the bedroom.
Lisa moved to the washroom from the bedroom.

Question: Where is Lisa?
Ground truth: washroom

Prediction:
✗ RNN: bathroom
✓ RNN-PG: washroom
✓ DMN-PG: washroom

Task 8: Where was [person] after [place]?

Story: · · ·
Jennifer read the book in the bedroom.
Jennifer looked at the wristwatch in the bedroom.
Jennifer stood up from the bed in the bedroom.
Jennifer turned off the light in the bedroom.
Jennifer moved to the bathroom from the bedroom.
Jennifer turned on the light in the bathroom.
· · ·
Jennifer washed her hands in the bathroom.
Jennifer turned off the water in the bathroom.
Jennifer wiped her hands with the towel in the bathroom.
Jennifer turned off the light in the bathroom.
Jennifer moved to the living room from the bathroom.
Jennifer mopped the floor with the mop in the living room.
Jennifer moved to the kitchen from the living room.
Jennifer turned on the water in the kitchen.
· · ·

Question: Where was Jennifer after the bedroom?
Ground truth: bathroom

Prediction:
✗ RNN: kitchen
✗ RNN-PG: living room
✓ DMN-PG: bathroom

Task 11: Who is in [place]?

Story: · · ·
David ate the cracker sand in the living room.
David drank coffee in the living room.
David turned off the television in the living room.
David turned off the air conditioner in the living room.
David stood up from the sofa in the living room.
David held the cup in the living room.
David held the plate in the living room.

Question: Who is in the living room?
Ground truth: David

Prediction:
✗ RNN: Sheldon
✓ RNN-PG: David
✓ DMN-PG: David

Task 13: Who [activity]?

Story: · · ·
Matt put the coffee bottle in the cupboard in the kitchen.
Matt put the cracker pack in the cupboard in the kitchen.
Matt closed the cupboard in the kitchen.
Matt turned off the electric kettle in the kitchen.
Matt poured hot water into the cup in the kitchen.
Matt poured off the hot water in the kitchen sink in the kitchen.
Matt stirred the coffee with the spoon in the kitchen.

Question: Who closed the cupboard?
Ground truth: Matt

Prediction:
✗ RNN: Howard
✓ RNN-PG: Matt
✓ DMN-PG: Matt

Task 16: Which rooms did [person] go?

Story:
Courteney removed the shoes in the entrance.
Courteney put on the slippers in the entrance.
Courteney moved to the bedroom from the entrance.
Courteney turned on the light in the bedroom.
Courteney sat down on the bed in the bedroom.
Courteney wiped the table with the duster in the bedroom.
Courteney read the book in the bedroom.
· · ·
Courteney stood up from the bed in the bedroom.
Courteney turned off the light in the bedroom.
Courteney moved to the living room from the bedroom.
Courteney turned on the light in the living room.
Courteney moved to the kitchen from the living room.
Courteney turned on the water in the kitchen.
Courteney washed her hands in the kitchen.
· · ·

Question: Which rooms did Courteney go?
Ground truth: bedroom, entrance, kitchen, living room

Prediction:
✗ RNN: bedroom, kitchen, living room
✗ RNN-PG: bedroom, kitchen, living room
✓ DMN-PG: bedroom, entrance, kitchen, living room

Task 17: What did [person] do just now?

Story: · · ·
Matt held the coffee bottle in the kitchen.
Matt held the cracker pack in the kitchen.
Matt put the coffee bottle in the cupboard in the kitchen.
Matt put the cracker pack in the cupboard in the kitchen.
Matt closed the cupboard in the kitchen.
Matt turned off the electric kettle in the kitchen.
Matt poured hot water into the cup in the kitchen.

Question: What did Matt do just now?
Ground truth: poured hot water into the cup

Prediction:
✗ RNN: closed the ice cream maker
✓ RNN-PG: poured hot water into the cup
✓ DMN-PG: poured hot water into the cup

FIGURE 7. Comparison of Sim2RealQA results of RNN, RNN-PG, and DMN-PG.

compared to the event-level attention in DMN-PG. These findings suggest that the integration of the pointer-generator decoder and event-level attention is effective for further improving our Sim2RealQA framework.

2) How well can models trained with the virtual-world data be generalized to the real-world data?

To assess the generalization of our proposed Sim2RealQA framework, we compared the methods, which used only a virtual-world dataset for training, to the methods trained on the target, which used a real-world dataset. Those trained on the target results revealed ideal performance, but the target real-world answer labels are actually unavailable. By comparing the *Sim2RealQA* and ideal *Train on target* performances, we can quantitatively investigate how well models trained with the virtual-world data generalize to the real-world data. The results are presented in Fig. 8 (left). Horizontal lines show the Sim2RealQA performance because it did not use any target examples for training. As we anticipated, the performance of the methods trained on the target

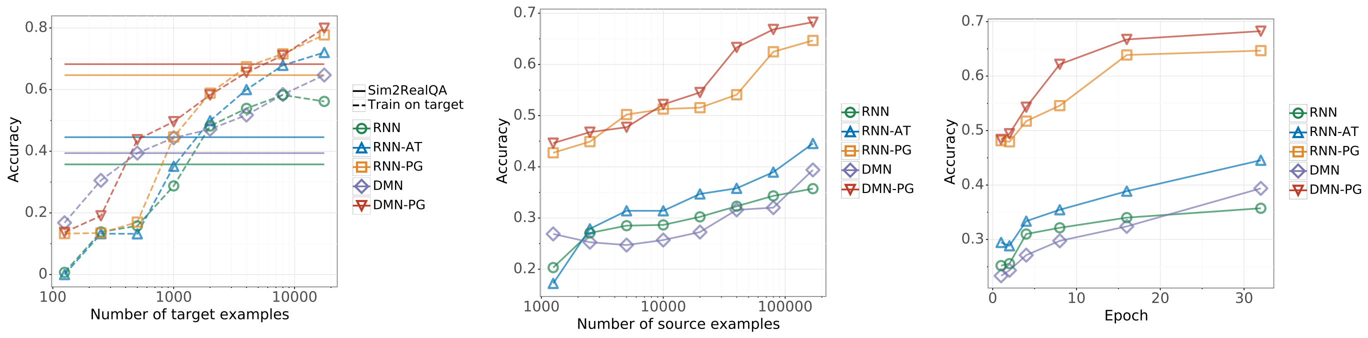


FIGURE 8. Sim2RealQA performance of baselines and proposed model: Sim2RealQA performance and oracle supervised QA performance over a number of examples in real-world dataset used for training (**left**); Sim2RealQA performance over number of examples in virtual-world dataset used for training (**center**); learning curves and accuracies (**right**).

improved with an increase in the proportion of real-world examples. Compared to the oracle methods that were trained with all the target examples, the methods using Sim2RealQA still have room for improvement. However, these oracle methods achieved lower accuracy in cases that involved an inadequate amount of training data because the performance of the neural QA models relies heavily on many labeled training datasets. Note that all the RNN, RNN-AT, RNN-PG, DMN, and DMN-PG methods on the Sim2RealQA framework outperformed the oracle methods with a small real-world training dataset. The DMN-PG of Sim2RealQA significantly outperformed all the methods trained with a target of 1,000 examples, a number that we cannot realistically collect. The result indicates the effectiveness of the proposed Sim2RealQA framework in the absence of real-world answers. This finding is useful because making real-world QA datasets is extremely difficult and laborious due to privacy reasons.

3) Does generalization to real-world data improve with more virtual-world data?

To validate the generalization ability of the proposed methods when the training set size increases, we explored the models' performances with the virtual-world QA datasets of several training data sizes. In Fig. 8 (center), the RNN, RNN-AT, RNN-PG, DMN, and DMN-PG performances steadily improved as the virtual-world training data size increased. DMN-PG outperformed the other methods when using a large amount of examples for training. The results suggest that using a large amount of virtual-world QA datasets is effective for more accurately solving real-world QA problems. This finding is fruitful because we can obtain diverse daily life stories from simulators and compile a large amount of virtual-world QA datasets without breaching privacy.

4) Do the pointer-generator mechanism and event-level attention quickly improve the Sim2RealQA performance?

Next we studied how the number of training epochs affected the Sim2RealQA performance. Fig. 8 (right) shows the accuracy of RNN, RNN-AT, RNN-PG, DMN, and DMN-PG over the {1, 2, 4, 8, 16, 32} epochs with the Sim2RealQA framework. With an increase of the training epochs, the performance of all the QA methods also improved. In addition, the models with a pointer-generator decoder (DMN-PG and RNN-PG) dramatically outperformed those without it (RNN, RNN-AT, and DMN), despite many fewer training epochs. DMN-PG outperformed the other methods over all the training epochs, indicating the pointer-generator mechanism's effectiveness for quick learning. A combination of pointer-generator decoder and relevance matching between question and story events (i.e., event level attention) more quickly improved the generalization to the real-world data.

VI. CONCLUSION

We proposed a novel simulation to a real QA (Sim2RealQA) framework that trains a neural QA model with many QA datasets produced in a life simulator and used it for solving real-word QA problems. To evaluate our framework, we developed real-world and virtual-world QA datasets using an actual house and a pre-made life simulator. We validated our proposed approach with a neural QA model that can address different entities between both worlds by combining the pointer-generator decoder with relevance matching between question and story events. From the experiments, we found that our method accurately solved real-world QA problems with the aid of virtual-world QA datasets. Moreover, our model, which was completely trained with the virtual-world QA dataset, significantly outperformed models trained with 1,000 examples in a target domain. In addition, the Sim2RealQA performance improved with an increasing number of examples from virtual-world QA datasets that can be created while protecting privacy. Furthermore, Sim2RealQA's quick learning was achieved by the integration of a pointer-generator mechanism and relevance matching (i.e., event-level attention). These findings support that using life simulations is a promising approach for solving real-world QA problems when no real-world answers are available.

In future work, we will refine our model to detect the temporal intention of the user's question and answer questions about events that occurred at specific dates and times in a long daily life. In this study, we created textual questions with the template-based approach to purely investigate the models trained with virtual-world data and generalize to real-world data with fewer noise settings. In more realistic situations, natural language questions are composed by a person seeking complicated real-world information, which offer multiple ways to say the same things. Another aspect of our future work will create datasets that include more natural and complex questions. Such diverse questions will be beneficial for robust and accurate real-world QA answering.

ACKNOWLEDGMENTS

This work was supported by JST ACT-I Grant Number JPMJPR18UT, JST CREST Grant Number JPMJCR15E2, and JSPS KAKENHI Grant Number JP18KK0284.

REFERENCES

[1] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” in Pervasive, 2004, pp. 1–17.
[2] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, “SoundSense: scalable sound sensing for people-centric applications on mobile phones,” in MobiSys, 2009, pp. 165–178.
[3] P. Lukowicz, J. A. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner, “Recognizing workshop activity using body worn microphones and accelerometers,” in Pervasive, 2004, pp. 18–32.
[4] T. Maekawa and S. Watanabe, “Unsupervised activity recognition with user’s physical characteristics data,” in International Symposium on Wearable Computers (ISWC), 2011, pp. 89–96.
[5] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Ishiguro, K. Kamei, Y. Sakurai, and T. Okadome, “Object-based activity recognition with heterogeneous sensors on wrist,” in Pervasive, 2010, pp. 246–264.
[6] R. Want, A. Hopper, V. Falcão, and J. Gibbons, “The active badge location system,” ACM Transactions on Information Systems, vol. 10, no. 1, pp. 91–102, 1992.
[7] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The cricket location-support system,” in MobiCom, 2000, pp. 32–43.
[8] M. Fan, A. T. Adams, and K. N. Truong, “Public restroom detection on mobile phone via active probing,” in International Symposium on Wearable Computers (ISWC), 2014, pp. 27–34.
[9] Y.-C. Tung and K. G. Shin, “EchoTag: accurate infrastructure-free indoor location tagging with smartphones,” in MobiCom, 2015, pp. 525–536.
[10] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, “Bluetooth positioning using RSSI and triangulation methods,” in IEEE CCNC, 2013, pp. 837–842.
[11] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter et al., “Place lab: Device positioning using radio beacons in the wild,” in Pervasive, 2005, pp. 116–133.
[12] D. Taniuchi and T. Maekawa, “Robust Wi-Fi based indoor positioning with ensemble learning,” in WiMob, 2014, pp. 592–597.
[13] C. Gurrin, H. Joho, F. Hopfgartner, L. Zhou, V.-T. Ninh, T.-K. Le, R. Albatal, D.-T. Dang-Nguyen, and G. Healy, “Overview of the NTCIR-14 lifelog-3 task,” in NTCIR, 2019, pp. 14–26.
[14] C. Gurrin, A. F. Smeaton, and A. R. Doherty, “Lifelogging: Personal big data,” Foundations and Trends in Information Retrieval, vol. 8, no. 1, pp. 1–125, 2014.
[15] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood, “Sensecam: A retrospective memory aid,” in Ubicomp, 2006, pp. 177–193.
[16] F. Hopfgartner, C. Gurrin, and H. Joho, “Rethinking the test collection methodology for personal self-tracking data,” in MMM. Springer, 2020, pp. 463–474.
[17] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Kamei, Y. Sakurai, and T. Okadome, “Object-blog system for environment-generated content,” IEEE Pervasive Computing, vol. 7, no. 4, pp. 20–27, 2008.
[18] T. Miyanishi, J. Hirayama, A. Kanemura, and M. Kawanabe, “Answering mixed type questions about daily living episodes,” in IJCAI, 2018, pp. 4265–4271.
[19] F. Hill, A. Bordes, S. Chopra, and J. Weston, “The goldilocks principle: Reading children’s books with explicit memory representations,” in ICLR, 2016.
[20] T. Kočiský, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, “The NarrativeQA reading comprehension challenge,” Transactions of the Association for Computational Linguistics, vol. 6, pp. 317–328, 2018.
[21] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, “Towards AI-complete question answering: A set of prerequisite toy tasks,” in ICLR, 2016.
[22] T. Linjordet and K. Balog, “Impact of training dataset size on neural answer selection models,” in ECIR, 2019, pp. 828–835.
[23] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in ICCV, 2017, pp. 843–852.
[24] M. Blum, A. S. Pentland, and G. Tröster, “Insense: Interest-based life logging,” IEEE Multimedia, vol. 13, no. 4, pp. 40–48, 2006.
[25] J. Lester, T. Choudhury, and G. Borriello, “A practical approach to recognizing physical activities,” in Pervasive, 2006, pp. 1–16.
[26] P. Lukowicz, H. Junker, M. Stager, T. V. Buren, and G. Tröster, “Wearnet: A distributed multi-sensor system for context aware wearables,” in Ubicomp, 2002, pp. 361–370.
[27] D. Castro, S. Hickson, V. Bettadapura, E. Thomaz, G. Abowd, H. Christensen, and I. Essa, “Predicting daily activities from egocentric images using deep learning,” in International Symposium on Wearable Computers (ISWC), 2015, pp. 75–82.
[28] M. Ma, H. Fan, and K. M. Kitani, “Going deeper into first-person activity recognition,” in CVPR, 2016, pp. 1894–1903.
[29] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson, “Mining models of human activities from the web,” in WWW, 2004, pp. 573–582.
[30] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hähnel, “Inferring activities from interactions with objects,” IEEE Pervasive Computing, vol. 3, no. 4, pp. 50–57, 2004.
[31] E. M. Tapia, S. S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” in Pervasive, 2004, pp. 158–175.
[32] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, “Accurate activity recognition in a home setting,” in Ubicomp, 2008, pp. 1–9.
[33] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 37, no. 6, pp. 1067–1080, 2007.
[34] M. Tachikawa, T. Maekawa, and Y. Matsushita, “Predicting location semantics combining active and passive sensing with environment-independent classifier,” in UbiComp, 2016, pp. 220–231.
[35] M. Li, X. Zhu, and S. Gong, “Unsupervised tracklet person re-identification,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.
[36] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, “Vision-and-language navigation: interpreting visually-grounded navigation instructions in real environments,” in CVPR, 2018, pp. 3674–3683.
[37] W. Hao, C. Li, X. Li, L. Carin, and J. Gao, “Towards learning a generic agent for vision-and-language navigation via pre-training,” in CVPR, 2020.
[38] S. Kurita and K. Cho, “Generative language-grounded policy in vision-and-language navigation with bayes’ rule,” in ICLR, 2021.
[39] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi, “Mapping instructions to actions in 3D environments with visual goal prediction,” in EMNLP, 2018, pp. 2667–2678.
[40] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, “Embodied question answering,” in CVPR, 2018, pp. 1–10.
[41] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, “IQA: Visual question answering in interactive environments,” in CVPR, 2018, pp. 4089–4098.
[42] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, “Embodied question answering in photorealistic environments with point cloud perception,” in CVPR, 2019, pp. 6659–6668.
[43] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, “VirtualHome: Simulating household activities via programs,” in CVPR, 2018, pp. 8494–8502.
[44] X. Puig, T. Shu, S. Li, Z. Wang, Y.-H. Liao, J. B. Tenenbaum, S. Fidler, and A. Torralba, “Watch-and-help: A challenge for social perception and human-AI collaboration,” in ICLR, 2021.

[45] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in ICRA, 2017, pp. 3357–3364.

[46] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in CVPR, 2020.

[47] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V.-D. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," in ICRA, 2019, pp. 4818–4824.

[48] F. Sadeghi and S. Levine, "CAD2RL: real single-image flight without a single real image," in RSS, 2017.

[49] M. Andrychowicz, B. Baker, M. Chociej, R. JA3zefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," The International Journal of Robotics Research, vol. 39, no. 1, pp. 3–20, 2020.

[50] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: learning agile locomotion for quadruped robots," in RSS, 2018.

[51] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in ECCV, 2016, pp. 102–118.

[52] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in EMNLP, 2016, pp. 2383–2392.

[53] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in ACL, 2018, pp. 784–789.

[54] M. Richardson, C. J. Burges, and E. Renshaw, "MCTest: A challenge dataset for the open-domain machine comprehension of text," in EMNLP, 2013, pp. 193–203.

[55] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li, "Neural generative question answering," pp. 2972–2978, 2016.

[56] Y. Fu and Y. Feng, "Natural answer generation with heterogeneous memory," in NAACL-HLT, 2018, pp. 185–195.

[57] S. He, C. Liu, K. Liu, and J. Zhao, "Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning," in ACL, 2017, pp. 199–208.

[58] S. Reddy, D. Raghu, M. M. Khapra, and S. Joshi, "Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model," in EACL, 2017, pp. 376–385.

[59] B. Liu, H. Wei, D. Niu, H. Chen, and Y. He, "Asking questions the human way: Scalable question-answer generation from text corpus," in The Web Conference, 2020, p. 2032–2043.

[60] J. E. Weston, "Dialog-based language learning," in NeurIPS, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., 2016.

[61] J. Wu, X. Wang, and W. Y. Wang, "Self-supervised dialogue learning," in ACL, 2019, pp. 3857–3867.

[62] Y. Deng, Y. Shen, M. Yang, Y. Li, N. Du, W. Fan, and K. Lei, "Knowledge as a bridge: Improving cross-domain answer selection with external knowledge," in COLING, 2018, pp. 3295–3305.

[63] S. Min, M. Seo, and H. Hajishirzi, "Question answering through transfer learning from large fine-grained supervision data," in ACL, 2017, pp. 510–517.

[64] B. Dhingra, D. Danish, and D. Rajagopal, "Simple and effective semi-supervised question answering," in NAACL-HLT, 2018, pp. 582–587.

[65] Z. Yang, J. Hu, R. Salakhutdinov, and W. Cohen, "Semi-supervised QA with generative domain-adaptive nets," in ACL, 2017, pp. 1040–1050.

[66] Y.-A. Chung, H.-y. Lee, and J. Glass, "Supervised and unsupervised transfer learning for question answering," in NAACL-HLT, 2018, pp. 1585–1594.

[67] D. Golub, P.-S. Huang, X. He, and L. Deng, "Two-stage synthesis networks for transfer learning in machine comprehension," in EMNLP, 2017, pp. 835–844.

[68] K. Sun, D. Yu, D. Yu, and C. Cardie, "Improving machine reading comprehension with general reading strategies," in ACL, 2019, pp. 2633–2643.

[69] A. Talmor and J. Berant, "MultiQA: An empirical investigation of generalization and transfer in reading comprehension," in ACL, 2019, pp. 4911–4921.

[70] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," in 3DV, 2017.

[71] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in CVPR, 2018, pp. 9068–9079.

[72] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in CVPR, 2020, pp. 10 740–10 749.

[73] M. Shridhar, X. Yuan, M.-A. Cote, Y. Bisk, A. Trischler, and M. Hausknecht, "ALFWorld: Aligning text and embodied environments for interactive learning," in ICLR, 2021.

[74] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen, "A corpus and cloze evaluation for deeper understanding of commonsense stories," in NAACL-HLT, 2016, pp. 839–849.

[75] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "MovieQA: Understanding stories in movies through question-answering," in CVPR, 2016, pp. 4631–4640.

[76] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning," in CVPR, 2017, pp. 2902–2910.

[77] D. A. Hudson and C. D. Manning, "Gqa: A new dataset for real-world visual reasoning and compositional question answering," in CVPR, 2019, pp. 6700–6709.

[78] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in ICML, 2016, pp. 1378–1387.

[79] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in ICML, 2016, pp. 2397–2406.

[80] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in ACL, 2017, pp. 1073–1083.

[81] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in EMNLP, 2014, pp. 1532–1543.

[82] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in NeurIPS Workshop, 2014.

[83] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681, 1997.

[84] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in ICML, 2010, pp. 807–814.

[85] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in NeurIPS, 2015, pp. 2692–2700.

[86] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in ICLR, 2016.

[87] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in ACL, August 2016, pp. 140–149.

[88] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in NeurIPS, 2014, pp. 3104–3112.

[89] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in ICLR, 2015.

[90] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in EMNLP, 2015, pp. 1412–1421.

[91] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual question answering," in ICCV, 2015, pp. 2425–2433.

[92] A. Anand, E. Belilovsky, K. Kastner, H. Larochelle, and A. Courville, "Blindfold baselines for embodied QA," in ViGIL, 2018.

[93] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in ICLR, 2015.

[94] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[95] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in EMNLP, 2016, pp. 1400–1409.

TAIKI MIYANISHI received the B.S. degree from the Department of Computer Science and Systems Engineering, Kobe University in 2009, the M.S degree from the Graduate School of Engineering, Kobe University in 2011, and the Ph.D. degree from Graduate School of System Informatics, Kobe University, Hyogo Japan, in 2014.

Since 2014, he has been a Researcher in the Department of Dynamic Brain Imaging, Advanced Telecommunications and Research Institute International, Kyoto, Japan. His research interests include ubiquitous computing, natural language processing, and computer vision.

TAKUYA MAEKAWA is an Associate Professor in the Graduate School of Information Science and Technology, Osaka University. He received his bachelor degree from School of Engineering, Osaka University in 2003. In 2004, he received his master degree from Graduate School of Information Science and Technology, Osaka University. In 2006, he received his doctor degree (Information Science and Technology) from Graduate School of Information Science and Technology, Osaka University. After that he worked for NTT Communication Science Laboratory for six years. He was awarded the IPSJ/IEEE Computer Society Young Computer Researcher Award on the topic of zero-shot and few-shot unobtrusive context recognition for pervasive computing in 2019. His research interest includes sensor-based context recognition techniques for pervasive/ubiquitous computing and animal behavior understanding.

MOTOAKI KAWANABE received the Master's degree in 1992, and the Ph.D. degree in mathematical statistics in 1995, both from the Department of Mathematical Engineering, University of Tokyo, Tokyo, Japan. After receiving the Ph.D. degree, he became an Assistant Professor at the University of Tokyo. He joined the Fraunhofer Institute FIRST, Berlin, Germany in 2000 as a Senior Researcher. Until fall 2011, he led the group for the THESEUS project on image annotation and retrieval there. He is currently a department head of Advanced Telecommunications Research Institute International (ATR), and a team leader of the RIKEN Center for Advanced Intelligence Project (AIP), Kyoto, Japan. His research interests include computer vision, biomedical data analysis, statistical signal processing, and machine learning.