Neural-network-based Memory for a Social Robot: Learning a Memory Model of Human Behavior from Data

MALCOLM DOERING and TAKAYUKI KANDA, Kyoto University and ATR, Japan
HIROSHI ISHIGURO, ATR, Japan and Osaka University, Japan

Many recent studies have shown that behaviors and interaction logic for social robots can be learned automatically from natural examples of human-human interaction by machine learning algorithms, with minimal input from human designers [1–4]. In this work, we exceed the capabilities of the previous approaches by giving the robot *memory*. In earlier work, the robot's actions were decided based only on a narrow temporal window of the current interaction context. However, human behaviors often depend on more temporally distant events in the interaction history. Thus, we raise the question of whether (and how) an automated behavior learning system can learn a memory representation of interaction history within a simulated camera shop scenario. An analysis of the types of *memory-setting* and *memory-dependent* actions that occur in the camera shop scenario is presented. Then, to create more examples of such actions for evaluating a shopkeeper robot behavior learning system, an interaction dataset is simulated. A Gated Recurrent Unit (GRU) neural network architecture is applied in the behavior learning system, which learns a memory representation for performing memory-dependent actions. In an offline evaluation, the GRU system significantly outperformed a without-memory baseline system at generating appropriate memory-dependent actions. Finally, an analysis of the GRU architecture's memory representation is presented.

CCS Concepts: • **Human-centered computing** → *Human computer interaction (HCI)*; • **Computing methodologies** → *Learning from demonstrations*; *Neural networks*;

Additional Key Words and Phrases: Knowledge representation and reasoning, neural networks, learning from demonstrations, human computer interaction, robotics

ACM Reference format:

Malcolm Doering, Takayuki Kanda, and Hiroshi Ishiguro. 2019. Neural-network-based Memory for a Social Robot: Learning a Memory Model of Human Behavior from Data. *ACM Trans. Hum.-Robot Interact.* 8, 4, Article 24 (November 2019), 27 pages.
<https://doi.org/10.1145/3338810>

1 INTRODUCTION

Researchers have recently explored the role of social robots in a variety of domains, including elder care [6, 7], shop-keeping [1, 2, 4, 3, 55], hotels [8], education [9], travel [10], and as personal

This work was supported in by the JST ERATO Ishiguro Symbiotic Human-Robot Interaction Project, Grant Number JP-MJER1401.

Authors' addresses: M. Doering and T. Kanda, Kanda Laboratory, Department of Social Informatics, Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan; emails: doering@robot.soc.i.kyoto-u.ac.jp, kanda@i.kyoto-u.ac.jp; H. Ishiguro, Intelligent Robotics Laboratory, Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama Toyonaka Osaka 560-8531 Japan; email: ishiguro@irl.sys.es.osaka-u.ac.jp.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

2573-9522/2019/11-ART24

<https://doi.org/10.1145/3338810>

companions [11, 12]. As the capability of artificial intelligence increases, this trend is likely to continue. However, one of the greatest challenges to introducing social robots into new domains is the time-consuming, tedious nature of designing interaction behaviors for all the scenarios the robot may encounter.

In response to this challenge, previous works have proposed data-driven approaches for designing social robot behaviors automatically, with minimal input from a human designer, by imitation learning from examples of natural human-human interaction [1, 2, 4, 3, 10, 55]. While impressive, with these systems the robot is limited to responding to contextual information from within only a short temporal window of the overall interaction. But in reality, many interaction behaviors depend on remembering key information over longer time periods.

In general, humans are able to learn a great deal of information about their interaction partners through interaction, which helps them to adapt their behavior to improve the quality of interaction. The information that a person holds in memory about an interaction partner affects how they might explain concepts, describe events, or introduce the interaction partner to others. For example, consider the following interaction between a customer and a shopkeeper:

Shopkeeper: “What brings you to Osaka?”

Customer: “I’m here for work. I often travel between Tokyo and Osaka.”

<... Later in the interaction, the *Shopkeeper* introduces a new camera... >

Customer: “Do you have anything else available?”

Shopkeeper: “Since you often travel, the Nikon would suit your needs. It’s small and lightweight.”

In this example, the shopkeeper is not merely reacting to what the customer said immediately before; he has remembered some information about the customer, which was stated earlier, and made a recommendation based on that memory. Whether or not a robot can learn a memory model of an interaction partner like this purely from example interaction data is an open question. In this work, we propose a data-driven technique that enables a robot to automatically remember such useful information and act on it during interaction with a human.

In our approach, we analyzed the presence of *memory-setting* and *memory-dependent* actions, like those in the above example, in a dataset of human-human interactions in a camera shop scenario. We discovered that there is a wide diversity of these actions, which occur in various patterns of dependence. Since each pattern appears only rarely, it would be difficult for a machine learning algorithm to learn an accurate representation of the shopkeeper’s memory. Therefore, we simulated a larger number of interactions that contain memory-setting and memory-dependent actions, and subsequently used it to train and test our proposed learning system. The purpose of the proposed system is to learn the behaviors and interaction logic of the shopkeeper, including how to perform appropriate memory-dependent actions so the human shopkeeper can be replaced by a robot. It accomplishes this by using a recurrent neural network architecture with gated recurrent units, which we evaluated in an offline evaluation. Last, we present an analysis of how the neural network represents memories.

The research question we sought to answer with this work is *whether* it is possible to build a memory system for human-robot interaction in a data-driven way. To our knowledge, there have not been any previous data-driven, behavior-learning systems designed and evaluated for the purpose of modeling memories in human-robot interaction. The most similar work to our proposed method is context-sensitive recurrent neural architectures [13–15]. But, they have only been applied to unimodal interaction data (dialogs) without automatic speech recognition errors, which are common in HRI. Furthermore, we go a step further than training a testing a model (Section 6) by first analyzing the types of memory-dependent behaviors that occur in

human-human interaction and then conducting an analysis to illustrate *how* a gated recurrent neural network represents memories (Section 7).

2 RELATED WORKS

2.1 Memory for Social Robots

There are several fields of study related to modeling memory for human-robot interaction. **Long-term HRI** focuses on designing robots that can build positive relationships with individual humans over time periods encompassing multiple interactions [16–19]. In such interactions, the robot’s *long-term* memories of its previous interactions with individuals it has previously interacted with can help build positive relationships. In contrast, our work focuses on the use of robots in domains where the human and robot will not necessarily meet more than once, e.g., between a shopkeeper and a customer; thus, the primary challenge is to generate robot actions consistent with the context of the current interaction. Therefore, we concentrate on *short-term* memory and not *long-term* memory.

Cognitive architectures are models of human cognition that contain components such as *working memory* and *episodic memory* [20, 21], which are more similar to the *short-term* memory modeled in our approach. Cognitive architectures have also been used to control robots and to predict human actions in HRI [22]. But, this approach relies on hand-coded models based on first principles, which can be difficult to design. In contrast, the data-driven approach to designing robot behaviors and interaction logic can be automated so it requires less effort from designers.

User modeling is used in dialog systems, and more recently HRI, to model users’ beliefs, goals, and plans such that the system can adapt to individual users [23–26]. Typically, user models are manually defined by domain experts, i.e., *what* is to be remembered by the system must be specified prior to interaction. Conversely, our memory modeling approach is data-driven, wherein the contents of memory are learned automatically, without input from expensive domain experts.

2.2 Machine Learning Approaches to Memory

An agent’s decision of which action it should take given some interaction context is often formulated as a sequential decision making process. Recurrent neural networks (RNN) are a data-driven method for modeling sequential processes, where, for example, the input to the network can be an interaction history (a sequence of human and agent actions), and the output could be the agent’s next action [27]. Long Short Term Memories (LSTM), and the simplified variant Gated Recurrent Units (GRU), are more advanced versions of RNNs designed to “remember” important information for longer durations [28, 29]. Memory Networks, Dynamic Memory Networks, and Neural Turing Machines are three newer neural architectures for sequence modeling that store past inputs in a searchable memory buffer [30–32]. Our proposed system implements a GRU to decide the robot’s actions.

2.3 Data-driven Learning of Interaction Behaviors for Social Robots and Virtual Humans

Liu et al. introduced the concept of training a social robot from examples of natural human-human interaction without any input from a human designer [1, 2, 4, 3]. The approach consists of clustering the raw interaction data to find discrete common actions, and then training a supervised learner using the action cluster IDs as labels. These systems have proven effective, but the robot’s action is decided based on only a narrow temporal window of the current interaction state.

More recently, Doering et al. introduced a data-driven behavior learning system for an android travel agent that learns interaction logic in the form of human-readable interaction rules [10]. In contrast to the work by Liu et al., the system contains a model of the time-persistent topic of

conversation that helps to resolve ambiguous customer questions. However, the topics modeled there are at a *course* level, and only the most recent topic is considered in deciding the android's action. However, our system can respond to *specific* actions that occur anywhere in the interaction history.

Admoni and Scassellati introduced a system that uses supervised learning with hand-labeled data to recognize and generate non-verbal behaviors for a robot tutor [33]. Leite et al. introduced a “semi-supervised learning” game-playing robot that could query workers via Amazon Mechanical Turk for new lines of dialog whenever it anticipated a novel situation [34]. These methods rely on hand annotated data or input from human designers, so they are not appropriate for rapid development of robot behaviors to new domains.

There also exist data-driven systems for designing behaviors for virtual humans. The “Restaurant Game” was a virtual restaurant in which two human players, playing the roles of customer and waiter, could navigate around the restaurant and interact via text and predefined actions [35, 36]. “Plan networks” were automatically extracted from the interaction data, enabling the automation of each character. The “Mars Escape” game used a similar approach and went a step further by embodying the learned behaviors in a real robot [37–39]. In contrast, data collected in the real, physical world, which is incorporated into our interaction simulation, has the additional challenge of sensor noise, which is absent in virtual worlds.

2.4 Data-driven Dialog Systems

Data-driven approaches are also popular for training dialog systems, where RNNs (and variations such as LSTM [28] and GRU [29]) have become a common method for context-sensitive language generation [14, 15]. Tian et al. compared several neural network architectures that integrate dialog context for response generation and found that hierarchical sequence models with attention performed the best [40]. Furthermore, their results showed that models with context were able to generate longer, more meaningful, and diverse replies than models without.

In the recent Dialog Systems Technology Challenge, several methods were presented for autonomously learning dialog behaviors from unannotated interaction datasets of Twitter interactions and simulated restaurant recommendation interactions [41]. The most successful approaches used variants of RNNs with LSTM. Similarly, other RNN architectures for unsupervised, end-to-end learning of context-sensitive conversational models demonstrated on Twitter data have been proposed [15].

Our approach is inspired by the related works on data-driven dialog systems, but there are some differences. Their training and testing data are not multimodal and do not contain speech recognition errors, in contrast to our data. Furthermore, little analysis is offered of why LSTMs work so well in these related works, but we offer an analysis of memory in interaction that motivates our network architecture.

The FRAMES corpus consists of manually annotated dialog transcripts in the travel agent domain collected in a Wizard-of-Oz setting and was designed for “adding memory to goal-oriented dialog systems” [42]. Shultz et al. present a system trained on this dataset that can remember which travel options were discussed so the agent can aid the customer in comparing options [43]. In contrast, in our work, we propose a system that learns to remember useful information without manually annotated labels.

Janarthanam and Lemon presented a finite-state-based dialog system for setting up home broadband networks that learned to adapt to users with different levels of knowledge using reinforcement learning [44]. This is similar to our system in that it can remember customer information and use it to decide actions. However, in their work the user knowledge levels and system actions were manually predefined. In contrast, we present a system that automatically learns from interaction

Fig. 1. The camera shop setup contained three camera displays. Sensors on the ceiling were used for tracking human position, and smartphones carried by the participants were used to capture speech (Liu et al. 2017).

data the set of system actions, the action policy, and which information is relevant for adapting to the user.

3 ANALYSIS OF HUMAN-HUMAN INTERACTION DATA

3.1 The Human-human Interaction Dataset

The human-human interaction dataset from Reference [3] was used to discover how memory was used in interaction and to inspire the design of an interaction simulator.

3.1.1 Scenario. The scenario consisted of typical interactions in a camera shop domain. A camera shop environment was set up in an $8\text{ m} \times 11\text{ m}$ experiment space containing a service counter and three camera models (a Canon EOS 5D Mark III, a Sony Alpha a6000, and a Nikon Coolpix S2800) in different locations, as shown in Figure 1. Each interaction contained one shopkeeper participant and one customer participant.

Note, this camera shop scenario was intentionally simplified in comparison to a real camera shop interaction, which would contain many more than three cameras. In a real camera shop scenario there are numerous interaction patterns, and multiple examples of each pattern would be required for a system to learn appropriate robot behaviors. Therefore, it was necessary to constrain the interaction scenario to collect a sufficient quantity of data for training and testing the system. In the future it will be possible to passively collect large quantities of natural, human-human interaction data in the real world, but for now behavior learning systems must be trained and tested in simplified scenarios.

3.1.2 Sensors. The participants' speech and position data was recorded automatically as they interacted with each other. A sensor network with a human position tracking system recorded the participants' location data [45].

To collect speech data, participants were required to use a handheld smartphone with an attached head-mounted microphone connected to the Google Speech API. To start and stop recording their speech a participant could touch anywhere on the smartphone's screen and an audible cue would play. This setup enabled participants to speak and operate the smartphone without distraction.

3.1.3 Participants. A total of 18 fluent English-speaking participants, with varied levels of knowledge about cameras, were recruited to play the role of the customer. To obtain a diverse

set of behaviors, two different participants, with different interaction styles, were chosen to play the role of the shopkeeper.

3.1.4 Procedure. The participants were encouraged to act naturally and to focus discussion on the features listed on camera spec. sheets (8 to 10 features per camera). Customer participants were encouraged to play with the cameras, browse the shop, or ask camera-related questions. The shopkeeper participants were instructed to wait at the service counter at the start of the interaction, be polite, and behave according to their role (e.g., greetings and farewells, letting the customer browse, answering questions, or introducing products when appropriate).

3.1.5 Summary of Collected Data. In total, 405 interactions were collected, containing 4,966 turns. Moreover, the data contained 4,061 shopkeeper utterances and 4,115 customer utterances. Several natural variations of phrasing (e.g., backchannels such as, “okay” and “I see, and...”) and terminology are present in the dataset [3]. Therefore, we consider the behaviors in the dataset to be realistic.

The dataset contained many automatic speech recognition errors. Of 400 utterances from a dataset collected in the same environment and scenario, and using the same equipment and recording methods, 53% were correctly recognized, 30% had minor errors (e.g., “can it should video” instead of “can it shoot video”), and 17% were complete nonsense [1]. The presence of sensor errors is one of the challenges of data-driven HRI; therefore, these ASR errors remain in the datasets.

3.2 Analysis of Memory-dependent Actions

To discover how memories of events during interaction were used by the participants, we analyzed the actions in the human-human dataset.

3.2.1 Defining Memory-dependent and Memory-setting Actions. Interactions in the dataset were discretized into sequences of alternating customer and shopkeeper *actions* [3]. Each *action* has a speech component (the participant’s utterance) and a spatial state component (the participant’s current location, movement departure location, movement arrival location, and a joint spatial formation—*waiting*, *face-to-face*, or *present object*). When exploring how memory was used in these interactions it helps to distinguish two special types of actions.

Some shopkeeper actions depend heavily on actions that occurred earlier in the interaction. A shopkeeper action that depends on an earlier action is called a **memory-dependent** action. The earlier action on which a memory-dependent action depends is called a **memory-setting** action. For example, if in the beginning of an interaction a customer states that they often travel, “I’m here for work. I often travel between Tokyo and Osaka” (a memory-setting action), then the shopkeeper should remember this. Then, later in the interaction the shopkeeper can make a recommendation, “Since you often travel, the Nikon would suit your needs. It’s small and lightweight” (a memory-dependent action). Throughout this work, we use the abbreviations *mem-dep* and *mem-set* for memory-dependent and memory-setting actions, respectively.

3.2.2 Types of Memory-dependent Actions. We identified eight types of memory-dependent shopkeeper actions in the human-human dataset. These are shown in Table 1. The leftmost column of Table 1 contains the names of the mem-dep action types; the second column contains the type of information in memory that each mem-dep action type depends on; the third column contains example mem-set utterances, which trigger the relevant information to be stored in memory; and the last column contains example sentences of each mem-dep action type.

We discovered memory-dependent actions that depend on the customer’s level of knowledge (*knowledge*), photography subject (*picture type*), and preferred camera features (*feature*). We also discovered mem-dep actions that continue the previous topic of conversation (*topic continue*). Some

Table 1. Types of Memory-dependent Actions Discovered from the Human-human Dataset

Mem-Dep type	Type of memory Mem-Dep depends on	Example Mem-Set utterance	Example Mem-Dep utterance
Knowledge	Customer's level of knowledge	"I don't really know much about cameras."	"It has automatic settings so you don't have to adjust anything."
Picture type	What customer wants to take pictures of	"I mostly take pictures at sporting events."	"For sport pictures it's better because it has 179 autofocus points, so it's really good for taking moving objects."
Feature	Customer's desired feature	"I'm looking for something cheap."	"It's a bit out of your budget; this one is \$550."
Topic continue	Continuation of previous topic	"This one has full manual settings?"	"But it also comes with manual control."
Additional info	Which information has already been presented	<shopkeeper has not yet introduced the camera's weight>	"It is also very light."
Previous camera	Previous camera of conversation	<shopkeeper was previously talking about the camera>	"The previous camera has a red color, so I recommend that for you."
Non-repeating	Action is not repeated	<shopkeeper has not yet greeted customer>	"Good afternoon, how can I help you?"
Other	Other	"I'm looking for a present for my father. He's not very good with technology though."	"Just remember, when thinking about your father, with this camera just point and shoot. You can use this right away."

mem-dep actions do not simply depend on memory of a single action. Actions in which the shopkeeper introduces a camera or feature (*additional info*) depend on memory of which cameras and features have already been talked about. Similarly, some mem-dep actions refer back to cameras that were previously under discussion (*previous camera*). Mem-dep actions that should not be repeated (*non-repeating*), such as greetings, depend on memory of whether the action has already been performed. Finally, some mem-dep actions depended on idiosyncratic characteristics of the participants or interaction (*other*).

3.2.3 Memory-dependent Action Statistics. After identifying the common mem-dep action types, we annotated each shopkeeper action with (1) the mem-dep type and (2) the index of the mem-set action on which it depended. We annotated 300 of the human-human interactions, which contained 3,666 turns. Table 2 contains the important information discovered from annotation.

The second and third columns from the left in Table 2 show the number of occurrences of each mem-dep type and the percent of the 3,666 annotated turns that are that type, respectively. With the exceptions of *topic continue* (7.4% of the data) and *non-repeating* (5.2% of the data), the mem-dep actions occur infrequently—the remaining six mem-dep types combined only constitute 6.6% percent of the data. The low number of examples suggests that it may be difficult for machine learning algorithms to learn to generate these types of actions.

The distance between a memory-setting action and a corresponding memory-dependent action depends on the memory-dependent action type. The last two columns display the average number of turns between each mem-dep action type and the mem-set actions on which they depend (and standard deviation). Note that the mem-dep types *knowledge* (average of 4.1 turns from mem-set, std. dev. 3.8), *picture type* (avg. 3.5, std. dev. 3.7), *previous camera* (avg. 4.7, std. dev. 3.6), and *other* (avg. 3.8, std. dev. 4.0) typically occur more than three turns after the mem-set action. In contrast,

Table 2. Occurrence Frequency of Memory-dependent Action Types and Distance from the Corresponding Memory-setting Actions

Mem-Dep type	Number of occurrences	Percent of the dataset	Ave. num. turns from Mem-Set (and std. dev.)
Knowledge	23	0.6%	4.1 (3.8)
Picture type	54	1.5%	3.5 (3.7)
Feature	24	0.7%	1.4 (1.9)
Topic continue	272	7.4%	0.5 (2.1)
Additional info.	66	1.8%	2.1 (2.4)
Previous camera	56	1.5%	4.7 (3.6)
Non-repeating	192	5.2%	N/A
Other	19	0.5%	3.8 (4.0)
Total	706	19.2%	

The numbers were computed from a subset of 300 human-human interactions containing 3,666 turns.

topic continue (avg. 0.5, std. dev. 2.1), *feature* (avg. 1.4, std. dev. 1.9), and *additional info.* (avg. 2.1, std. dev. 2.4) usually depend on much more local information. Since previous techniques have explored incorporation of local context in robot action prediction [2], in this work, we focus on actions like the former, which have long-term dependencies.

4 INTERACTION SIMULATION

The focus of this work is to discover whether a robot is capable of learning to represent memories so it can correctly perform actions that depend on those memories; however, the number of training examples in the human-human dataset is lacking. The human-human camera shop dataset described above contains a diverse set of memory-setting and memory-dependent actions that appear in various different patterns of dependence. Furthermore, each type of memory-setting and memory-dependent action can occur with many natural variations of utterance and may contain automatic speech recognition errors. Consequently, the number of examples of each pattern is lacking and thus difficult for a machine learning algorithm to learn.

We believe that in the future it is likely that much larger datasets of natural human-human interactions will be collected by sensor networks deployed in spaces where humans frequently interact. But in the meantime, we have developed an interaction simulator based on the interaction patterns observed in the human-human data to create a larger training dataset with more examples of memory-setting and memory-dependent actions.

4.1 Procedure

The simulator works by generating sequences of alternating shopkeeper and customer actions (Table 7 in the appendix). Moreover, in each generated interaction, the simulator uses action transition probabilities in combination with if-then style logic to generate contextually appropriate actions. Finally, customer and shopkeeper utterances from the human-human dataset are injected into the generated action sequences.

4.1.1 Action Transitions. Each simulated interaction starts with the customer at the door and the shopkeeper at the service counter. The customer then enters and approaches one of the three cameras or the shopkeeper. The shopkeeper may approach a browsing customer to assist them by asking their preferences, answering questions, or introducing the cameras and their features. In some cases the customer may state they simply wish to browse. Otherwise, the customer may ask about the cameras or state their preferences. The interaction proceeds until the customer decides

to leave the store. Throughout the interaction, the customer and shopkeeper's actions are partially decided by action transition probabilities.

The action transition probabilities were set manually based the interactions in the human-human dataset from Reference [3]. In that work, the customer's and shopkeeper's actions were segmented and discretized into sequences of alternating customer and shopkeeper actions. The transition probabilities were estimated based on observations of these sequences. For example, after the shopkeeper introduces a camera, there is a roughly 50% chance that the customer will remain silent (allowing the shopkeeper to take initiative) and a 50% chance that the customer will ask a question about a feature. In the future, it would be useful to compute the transition probabilities automatically by first clustering the human actions based on similarity (as in Section 5.3) and then counting the transitions between the *action clusters*.

4.1.2 If-then Logic for Contextually Appropriate Actions. For each interaction, the simulator modeled important information about the interaction history, including: whether the customer has stated they are just browsing and wish to be left alone, the current camera of conversation, a list of memory-setting actions that have been performed, a list of previous cameras of conversation, a list of previous features of conversation, and the customer and shopkeeper's locations.

Using if-then logic conditioned on these stored aspects of the interaction history, the simulator generates contextually appropriate actions. For example, the shopkeeper's actions are simulated such that he does not repeatedly approach browsing customers who wish to be left alone, remembers the preferences of the customer and makes appropriate recommendations, and remembers which cameras and features have already been talked about so he does not re-introduce them. Additionally, the customers' actions are simulated such that they do not ask about features or cameras that have already been introduced or asked about.

To generate memory-setting actions customized for different *customer types*, if-then logic was used. We designated 10 possible customer types based on the types of pictures they like to take and their level of expertise: *art school*, *simple everyday pictures*, *family and friends*, *nighttime*, *outdoor*, *pets*, *sports*, *travel*, *expert user*, and *novice user*. Whenever the customer performed a LOOKING_FOR_A_CAMERA_WITH_X action (from Table 7), X was determined based on the customer type, resulting in a customized memory-setting action.

To generate customized memory-dependent actions, if-then logic conditioned on the occurrence of the above memory-setting actions was used. Moreover, each customer type was associated with preferred camera features (Table 8 in the appendix). Using this knowledge, the shopkeeper was able to perform customized memory-dependent actions by appending phrases relating to the customer type to responses about their preferred features (e.g., "This one is very lightweight, *which makes it suitable for travel*."). (The cameras' features are shown in Table 9 of the appendix.)

4.1.3 Natural Variation and Speech Recognition Errors. In the final step of simulation, customer and shopkeeper utterances are assigned to each turn. For each action (from Table 7), a set of matching utterances was selected from the human-human dataset. The utterances from the human-human dataset contain natural variation in phrasing and also contain many speech recognition errors. By introducing these characteristics into the simulated dataset, it should present a similar learning challenge as the real human-human dataset.

4.2 The Simulated Dataset

In total, we simulated 50 K interactions, containing 619,848 customer-shopkeeper turns, for training and testing the proposed system. The data is available for download.¹ Table 3 presents an example simulated interaction.

¹<http://www.geminoid.jp/dataset/simulatedcamerashop/Dataset%20-%20Simulated%20camera%20shop.html>.

Table 3. An Example Simulated Interaction

Time Step	Person	Location	Utterance	Camera of conversation	Previous cameras and features of conversation	Memory	Action Type
1	C	DOOR					
	S	SERVICE COUNTER					
2	C	SONY					
	S	SONY	good afternoon how can I help today				
3	C	SONY	what cameras do you have help with that would be good for beginners			NOVICE	Mem-Set 'Novice user'
	S	NIKON	this kind of camera over here is fantastic it's like the next step along for using your mobile phone	NIKON	NIKON	"	
4	C	NIKON	okay let's see	NIKON	"	"	
	S	NIKON	this one is a point and shoot camera perfect for beginner user, you can just push the shutter button and you don't have to adjust anything it will auto adjust itself	NIKON	NIKON: point_and_shoot	"	Mem-Dep 'Novice user'
5	C	NIKON	huh	NIKON	"	"	
	S	NIKON	this camera is really good, it is only \$68, it's incredibly light, feel it up pick it up feel the weights	NIKON	NIKON: point_and_shoot, price	"	
⋮							
9	C	NIKON	okay thanks for the information		NIKON: point_and_shoot, price, ease_of_use, color, ...	"	
	S	NIKON	great thanks for your time have a great afternoon		"	"	
10	C	DOOR				"	
	S	SERVICE COUNTER				"	

5 THE DATA-DRIVEN MEMORY LEARNING SYSTEM

5.1 Overview

The goal of the system is to learn the behaviors of a shopkeeper from examples of human-human interaction, such that a robot can perform as a shopkeeper in interaction with real humans. Figure 2 shows an overview of the complete system that enables the robot shopkeeper to function.

A sensor network collects raw interaction data in the camera shop. It consists of an array of Microsoft Kinect sensors that track the human and robot's positions and a smartphone application that records the customer's speech. Automatic speech recognition is used to transcribe the customer's speech into text. The sensor data is sent to a behavior abstraction module that discretizes the raw tracking data into one of the typical stopping locations (door, middle, Canon, Sony, Nikon,

Fig. 2. Overview of the complete system for real-time interaction.

and service counter), which were discovered by clustering the stopping locations in the training interactions [1].

The interaction history stores all the customer and shopkeeper's previous actions (from turn 0 through $t-1$). When a new customer action is detected, it is appended to the interaction history. The actions are vectorized (Section 5.2) and fed into a neural network that outputs the robot's next action (Section 5.4), which also gets appended to the interaction history. In this way, a robot can perform the role of a shopkeeper in live interaction with a human.

In this work, we focused on the role of the GRU neural network in representing memories and enabling the robot to perform appropriate memory-dependent actions. Therefore, we trained the system on simulated interactions and tested it in an offline evaluation, rather than conducting live user studies. Similar versions of the complete system have been demonstrated to work in live interactions with real humans in previous work [1, 2].

5.2 Input Action Vectorization

The sequence of actions in an interaction must be vectorized before it can be input into the neural network. The input vectorization for a customer or shopkeeper action consists of a speech vector concatenated to a location vector.

Speech was represented using a binary bag-of-words representation, where each index of the speech vector represented a word in a vocabulary with a value of 1 if the word occurred in the utterance and 0 otherwise. Separate vocabularies were acquired for the customer and shopkeeper based on words spoken by customers and shopkeepers in the simulated dataset. The customer and shopkeeper vocabularies contained 938 and 1,255 words, respectively.

We opted for this simple speech representation for the purpose of reducing the input dimensionality (and thus the number of neural network parameters and training time) and increasing possibilities for interpretability. One limitation of bag-of-words representation is that word ordering information is lost. Previous works [1, 2, 3] used n -gram vectorization ($n = 1, 2$, and 3) with latent semantic analysis (LSA) for dimensionality reduction. However, LSA works by combining multiple n -grams into the same dimension, so interpretability is lost.

The location vector represented at which of the six locations (door, middle, Canon, Sony, Nikon, and service counter) the customer/shopkeeper was located. For these, we used a one-hot vector of five dimensions (since the customer never goes to the service counter and the shopkeeper never goes to the door).

The input vector for a single time step t consisted of the shopkeeper action vector s_{t-1} (1,255 speech and 5 location dimensions), and the customer action vector c_t (938 speech and 5 location dimensions) concatenated together, thus a vector of 2,203 dimensions.

5.3 Output Shopkeeper Action Clustering

The neural network works as a “sequence classifier,” which learns to output the most appropriate “class” for any input “sequence.” The output classes dictate the robot’s next action. To obtain these classes, the simulated shopkeeper actions were aggregated into clusters of similar actions, representing the set of common shopkeeper actions. To discretize the shopkeeper actions, each one was assigned the ID of its containing cluster. Shopkeeper actions were clustered by combining shopkeeper speech clusters with shopkeeper location information.

The goal of speech clustering was to find highly homogeneous clusters of semantically similar utterances. To do this, 25 K utterances were sampled from the simulated interactions, speech vectors and inter-utterance cosine distances were computed, and the Dynamic Tree Cut hierarchical clustering algorithm (DTC) was applied [46]. The clustering procedure yielded 1,083 speech clusters.

The DTC algorithm has several advantages for speech clustering. We experimented with several clustering algorithms and found DTC to consistently yield large, homogeneous clusters of semantically similar utterances. Furthermore, DTC automatically discovers the number of speech clusters from the data, which helps to automate the robot behavior learning process. Also, in contrast to traditional hierarchical text clustering methods [47], it cuts the tree structure at different heights depending on the degree of variation of utterances in the branches. This is suitable for speech data, since semantically similar utterances can have a wide range of variation depending on utterance length, natural variation of phrasing, and presence of ASR errors.

For the purpose of speech clustering, a separate speech vectorization model was used that encoded word ordering and emphasized important keywords. Each speech vector consisted of an n -gram vector ($n = 1, 2, \text{ and } 3$) concatenated to a keyword vector. Keywords were extracted by a cloud-based API (now part of IBM Watson²).

Speech clusters were combined with location information to yield the action clusters. More specifically, to get the shopkeeper’s action cluster ID at t , the ID of the speech cluster containing the shopkeeper’s utterance at t is combined with his current location (service counter, middle, Sony, Canon, or Nikon), yielding an action cluster ID. For example, if the shopkeeper says, “Thank you for coming,” which is in speech cluster 3, while standing near the *Sony*, then the action cluster ID is *3_Sony*. Combining the speech clusters with the shopkeeper location information yielded 1,479 shopkeeper action clusters.

Since the output robot action class must dictate the robot’s speech, a *typical utterance* was automatically selected for each speech cluster by finding the utterance that was most similar to all the other utterances in the speech cluster (medoid). The Levenshtein distance metric normalized for the length of the utterances was used. Complete utterances with few ASR errors tended to share the most similarities with other utterances in the same cluster. Therefore, *typical utterances* tended to be well formed and easy to understand.

²<https://www.ibm.com/watson/services/natural-language-understanding/>.

Fig. 3. (a) The neural network architecture: s_{t-1} is the previous shopkeeper action (1,255 speech and 5 location dimensions), c_t is the current customer action (938 speech and 5 location dimensions), and s_t is the next shopkeeper action (1,479 shopkeeper action clusters), which dictates the robot's next action. (b) The internal details of the GRU layer [5]: h_{t-1} and h_t are the previous and current GRU states, r_t is the reset gate output, z_t is the update gate output, and $\tilde{h}_{t-1}$ is the candidate GRU state.

5.4 Recurrent Neural Network Architecture

The GRU recurrent neural network takes a vectorized input sequence $s_0, c_1, \dots, s_{t-1}, c_t$ as input and outputs a probability distribution over the possible next shopkeeper actions, s_t . The neural network architecture consists of six layers, as shown in Figure 3(a). The first layer is the input layer, which takes as input each of the $s_{t'-1}, c_{t'}$ vectorizations for $0 < t' \leq t$ (that is, all actions from the beginning of the interaction up through the most recent customer action). The next three layers are dense hidden layers of 800 units each with leaky rectifier activation functions whose purpose is to condense the input into a lower dimensional encoding. The next layer is the gated recurrent unit (GRU) layer, as defined in Reference [29]. Finally, the last layer is the output layer, which represents the robot's next action.

The GRU layer is the most critical part of the architecture for learning memories of important events in the interaction. It does this by maintaining a state vector, h_t , over multiple time steps. The goal of training the network is for the GRU to learn to represent the occurrence of memory-setting actions within its state vector, such that the network can output appropriate memory-dependent actions.

The behavior of the GRU layer is defined with the following four equations:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \quad (1)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \quad (2)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b), \quad (3)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \quad (4)$$

where z_t is the update gate output, r_t is the reset gate output, $\tilde{h}_t$ is the candidate GRU state, h_t is the new GRU state (and output), and x_t is the current input (the output of the third dense hidden layer). The goal of r_t is to learn when to add information from the previous state h_{t-1} to the candidate state $\tilde{h}_t$ vs. when to only add information from the current input x_t to the candidate

state $\tilde{h}_t$. The goal of z_t is to learn when to keep the previous state h_{t-1} vs. when to overwrite it with the candidate state $\tilde{h}_t$. W_z , W_r , W , b_z , b_r , and b are weight and bias parameters that are learned during training. In total, the weights and biases constitute 8,071,879 parameters.

6 OFFLINE SYSTEM EVALUATION

To understand the role of the GRU in representing memories, we conducted an offline evaluation comparing the proposed data-driven memory system to a system without a GRU layer (i.e., memory). By comparing the proposed system to a baseline that is the same in all respects except for the lack of a GRU layer, the effects of the GRU layer on the system's performance can be isolated.

6.1 Experimental Conditions

The focus of this study was to develop a technique for generating appropriate memory-dependent actions by remembering relevant memory-setting actions that occur during interaction. Therefore, the experimental design compared the behaviors of two action prediction conditions.

No memory perceptron (NMP) consisted of a five-layer, feedforward neural network. The first layer is an input layer that takes the vectorization of the most recent customer action, the next three layers are dense hidden layers with 800 dimensions, and the last layer is a softmax layer of 1,479 dimensions, representing a probability distribution over the possible next shopkeeper actions. This architecture is similar to the proposed memory system, but without the GRU layer. Thus, it cannot "remember" past information. By comparing the proposed system to the NMP, we can discover how the GRU layer, and thus "memory," affects performance.

Proposed system consisted of the complete system introduced in the previous section.

6.2 Training the Systems

The two action predictors were trained using the same procedure. The first 75% of the simulated dataset (containing 464,612 customer input/shopkeeper output interaction turns) was used for training, and the first 500 turns (39 interactions) from the remaining 25% was used for testing. Categorical cross-entropy was used to compute the loss function. To encourage the networks to output memory-dependent actions, which occurred infrequently in the training data, the loss for each output action was weighted inversely proportional to the action's number of occurrences in the training data. The Adam optimizer was used to update the parameter values over 200 epochs [48].

6.3 Evaluation Procedure

One human evaluator, an internally recruited fluent English speaker (F, age 37) blind to the experimental conditions, evaluated each of the 500 predicted robot actions with a binary label of either *correct* or *incorrect*. To receive a *correct* label the robot's action must have provided the correct information. If the customer did not request any specific information, then the criterion was that the robot's action must be socially appropriate. In the case that the evaluator was unable to determine if the predicted action was correct or not, the instance was not included in the evaluation (0.2% of instances). Evaluations by a second evaluator, an internally recruited fluent English speaker (M, age 28) blind to the experimental conditions, showed a high degree of agreement, with a kappa coefficient of 0.63, so the evaluations were judged to be reliable.

To better understand the role of memory in outputting memory-dependent actions, we narrowed our focus to the *memory-dependent* actions that depend on *customer type* (described in Section 4.1.2). The 500 action predictions were divided into *customer-type-memory-dependent* and *other* subsets. If a robot action contained an utterance from a simulated action that depended on

Table 4. Percent of the *Proposed* and *No Memory Perceptron* systems' actions judged correct by a human evaluator in the offline evaluation

Condition	Overall Correct	Customer-type-mem-dep Correct	Other Correct
No Memory Perceptron	70.3% (N = 499)	13.6% (N = 22)	73.0% (N = 477)
Proposed System	85.8% (N = 499)	66.7% (N = 42)	87.5% (N = 457)
Chi Square Test	$p < .001$, $\chi^2 = 34.68$	$p < .001$, $\chi^2 = 16.26$	$p < .001$, $\chi^2 = 31.07$

Customer-type-mem-dep refers to the subset of output actions that depend on *memory of customer type*. *Other* refers to the remaining output actions. *Overall* is the both of these subsets combined. *N* refers to the number of actions that were evaluated for each system/subset. The last row contains the Chi Square test results comparing the two systems on each action subset. The proposed system performed significantly better on all action subsets.

memory of customer type, then it was labeled *customer-type-memory-dependent*. Otherwise, it was labeled *other*.

6.4 Results

The evaluation results and statistical analysis results are shown in Table 4. Overall, the proposed system performed the best with 85.8% action predictions judged correct. This was significantly better than the no memory perceptron (NMP) with 70.3% correct ($\chi^2(1) = 34.68$, $p < .001$).

On the *customer-type-memory-dependent* subset of action predictions, the proposed system had 66.7% action predictions correct. This was significantly better than the NMP with 13.6% correct ($\chi^2(1) = 16.26$, $p < .001$).

On the *other* subset of action predictions, the proposed system had 87.5% correct. This was significantly better than the NMP with 73.0% correct ($\chi^2(1) = 31.07$, $p < .001$).

Last, the proposed system output almost twice as many *customer-type-memory-dependent* actions as the NMP (42 actions vs. 22 actions, respectively) ($\chi^2(1) = 6.68$, $p = .010$).

6.5 Analysis of Incorrect Actions

We analyzed the proposed system's 14 incorrect *customer-type-memory-dependent* actions to understand the cause of the system's mistakes: 36% were incorrect because of inappropriate repetition, 36% because the robot did not provide the requested information, 14% because the robot's speech did not match the customer type (e.g., recommending a camera for simple, everyday photos to someone who wants to do outdoor photography), and 14% because the system did not distinguish between customer subtypes (e.g., customers who owned cats vs. those who owned dogs).

Furthermore, analysis revealed that the proposed system inappropriately repeated utterances less frequently than the NMP: 1.4% (7 instances) of the proposed system's actions were inappropriately repeated, while 4.4% (22 instances) of the NMPs were inappropriately repeated ($\chi^2(1) = 7.99$, $p = .004$). This demonstrates the importance of memory in avoiding repetition.

6.6 DISCUSSION

6.6.1 The Importance of Memory. The proposed system had some ability to "remember" the interaction history: It learned to correctly output *customer-type-memory-dependent* actions with 66.7% accuracy. However, the no memory perceptron (NMP) only output *customer-type-memory-dependent* actions correctly with 13.6% accuracy, since it had no knowledge of which memory-setting actions had occurred. This demonstrates the value of learning a memory representation for human-robot interaction.

The proposed system also performed better than NMP on the *other* subset of actions (87.5% vs. 73.0%). We observed that many of the NMP's mistakes resulted from (1) the robot repeatedly asking the customer what their preferences were, (2) re-greeting a customer who stated they want to browse and be left alone, and (3) reintroducing camera features that the robot had already introduced. To behave appropriately in these situations the robot needs to remember the interaction history.

6.6.2 Generalization and Overfitting. The proposed system has more learned parameters than training data, but this did not result in overfitting. The proposed system has about 17.4 times as many parameters as training data (8,071,879 parameters and 464,612 training points), but this is common practice with neural networks. Related work using neural networks in domains conceptually related to robot behavior learning, including machine translation [49, 50], dialog modeling [14, 15], and question answering [30, 51], present neural networks with in the order of 10 and 100 times as many parameters as training data.

The proposed system was able to generalize from the training set to the testing set, as demonstrated by the proposed system's high accuracy on the testing dataset (85.8% correct overall). To further enhance this point, here, we show that the simulated testing interactions were distinct from the simulated training interactions.

Analysis showed that the 39 testing interactions were indeed distinct from the training interactions. The normalized Levenshtein distance metric (NLD) was used to compute the differences between the 39 testing interactions and the training interactions. To compute the NLD between a testing interaction and a training interaction, first each unique combination of utterance and location was assigned a unique ID. Then, each interaction was represented as a sequence of these IDs. Finally, the NLD was computed between the two ID sequences. The average, minimum, and maximum distances between a testing interaction and a training interaction were 0.83 (std. dev. 0.07), 0.33, and 0.97, respectively. Thus, the testing interactions were distinct from the training interactions. In conclusion, the proposed system generalizes without overfitting.

7 ANALYSIS OF THE TRAINED NEURAL NETWORK

An analysis of the trained GRU model was conducted to elucidate how it represented memories in the camera shop scenario. The GRU layer of the neural network consisted of 800 units like that shown in Figure 3(b). Each of these units were trained to store relevant information from the input customer and shopkeeper actions over multiple time steps such that the network could output the most appropriate shopkeeper actions. The goal of the analysis is to examine the behavior in the GRU units that represent "memories." That is, which customer/shopkeeper actions are stored in these GRU units? And, what is the behavior of the GRU gates and output activations of these units?

First, the GRU units that are most likely to encode memories are found by computing their mutual information with ground truth memory labels. Second, we explain how treating GRU gate output values as binary values yields a simplified interpretation of their behavior that can be easily visualized. Last, the first two steps are combined to visualize the behavior of the memory encoding GRU units over time in two of the simulated interactions.

7.1 Finding GRU Dimensions that Encode Memories

To decide which of the 800 GRU units would be most interesting to examine in detail, the mutual information scores (MI) between ground truth memory labels and GRU units were computed from the training data [52]. MI aids in understanding the statistical relationships between certain GRU units and memory types based on aggregated data. The intuitive interpretation of MI is the amount of information given about one variable (e.g., the occurrence of a memory label) when some other

Table 5. GRU Units that Give the Maximum Amount of Mutual Information (MI) for Each Memory type

Memory Label	Art school	Everyday photos	Expert user	Family and friends	Nighttime	Novice user	Outdoor	Pets	Sports	Travel
GRU unit ID	305	77	601	45	517	524	218	152	363	462
MI (bits)	0.116	0.072	0.099	0.080	0.089	0.057	0.048	0.061	0.051	0.061

MI aids in understanding the statistical relationships between the GRU units and memory types based on aggregated data. Intuitively, MI indicates the amount of information that knowing the GRU state activation value gives about the occurrence of a memory label (and vice versa).

variable is known (e.g., a GRU's activation value). Here, we measure MI in bits. At least 1 bit of information is necessary to know whether a memory-setting action occurred (since it has either occurred or it has not). GRU units that have high mutual information with memory labels are critical for encoding those memories. We refer to these as “memory units.”

Typically MI is computed between two discrete variables; however, in the case of computing MI between memory types and GRU activations, the former variable is discrete and the latter is continuous. Therefore, the method for computing MI between discrete and continuous variables from Reference [53] was employed, with the parameter k set to 3. This method works by using a k -nearest-neighbors-based algorithm to compute a probability density function of the continuous variable and then computes MI based on that.

The ground truth memory labels were obtained from the interaction simulator. There were 10 possible memory labels in total, corresponding to the 10 possible *customer types* (knowledge level, preferred subject of photography, etc.) (Table 8). To generate the labels, all 10 memory labels were initialized to 0 in the beginning of each interaction. When the customer performed a memory-setting action, e.g., stated their preferred picture type, the appropriate memory label was set to 1 for the remaining time steps in the interaction. In this way, the GRU outputs, h_t , at each time step of each interaction can be compared to the corresponding ground truth memory labels at the same time step of the same interaction.

The GRU units with the maximum mutual information for each of the 10 memory labels are show in Table 5.

7.2 Interpretation of Binary GRU Gate Values

To explore the activity of the “memory units,” the output values of the GRU update and reset gates (z_t and r_t , respectively) were analyzed in addition to the output values of the GRU units themselves (h_t).

The output values of the update and reset gates were in the range (0, 1), since they are determined by a sigmoid function. Moreover, the sigmoid function tends to push the gate outputs to either 0 or 1. Indeed, the histograms of the update and reset gate output values for all time steps for all interactions of the training data (Figure 4) show that the values are usually close to either 0 or 1. Only 1.5% of the update gate outputs and 0.6% of the reset gate outputs did not fall into either the first or the last bin of the histogram. This allows us to treat the gate outputs as if they were binary.

Treating the gate output values as if they were binary greatly simplifies the interpretation of the GRU's activity. With two gates and binary values, there are four possible states that the gates

Fig. 4. Histograms of the GRU’s update (left) and reset (right) gate output values from the training data show that the gate values almost always fall into the first or last bin. That is, they are usually close to 0 or 1. Therefore, the gate values can be treated as binary values for this data.

Table 6. Interpretation of the binary GRU gate outputs

Update value	Reset value	Function in GRU	Interpretation
0	1	Previous state h_{t-1} becomes new state h_t (no state change).	<i>Preserve memory</i>
0	0		
1	1	Current input x_t and previous state h_{t-1} are combined to create new state h_t .	<i>Augment memory</i>
1	0	Previous state h_{t-1} is discarded and new state h_{t-1} is computed from current input x_t .	<i>Overwrite memory</i>

The leftmost column contains the binary values of a GRU’s *update* (z_t) and *reset* gates (r_t), the middle column describes the function of the gates when they have certain values, and the last column contains how the gate functions are interpreted when considering *memory* in interaction.

can be in. Actually, two of the states have the same function, so the gates can be thought of as being in one of three possible states at each time step of an interaction, as shown in Table 6. The three possible states of the gates are *preserve memory*, *augment memory*, and *overwrite memory*. The functions of these states are shown in Table 6.

This GRU analysis method could be applied to any problem where the gate outputs tend to be binary (i.e., close to 0 or 1). Theoretically, the gate outputs can be anywhere in the range (0, 1) (i.e., non-binary). However, for the purpose of interpretability, training techniques for pushing the gate outputs to either 0 or 1 have been proposed [54]. Therefore, our GRU analysis method could be applied to a wide variety of problems by using such training techniques.

7.3 Memory Visualization

With the binary gate interpretations, it is possible to visualize the GRU output activations, h_t , over the course of the interaction in a legible way. Two visualizations are shown in Figure 5.

The two grids in Figure 5 display data from two of the simulated training interactions. Short interactions of the same length were chosen to easily compare the behaviors of the memory units for two different interactions.

In the top interaction, the customer states that they are looking for a camera suitable for an expert user (c_t at $t = 2$). The network recognizes the occurrence of a memory-setting action based on the words in the customer’s statement (e.g., “looking for,” and “expert”). Later in the interaction, the proposed system (shopkeeper), having remembered the customer’s preference, is able to

Fig. 5. Visualizations of the GRU units most important for encoding memories in two interactions. s_{t-1} and c_t are the previous shopkeeper action and current customer action (inputs to the neural network). Predicted s_t is the predicted next robot action (output from the neural network). The time steps with memory-setting actions are highlighted in yellow. Each grid cell represents the internal state of a memory encoding GRU unit at the specified time step. The arrows represent the state of the GRU unit's gates and the cell color represents the output activation (h_t), per the color-bar keys.

introduce some information about the Canon camera relating to expert users (predicted s_t at $t = 3$). Similarly, in the bottom interaction, the customer states that they are looking for a camera for photographing sports (c_t at $t = 2$) and later the proposed system, having remembered, is able to refer to the customer's preference when introducing the autofocus feature (which helps take pictures

of moving subjects) (predicted s_t at $t = 3$). In both these examples, the proposed system learned to remember the memory-setting actions and later output appropriate memory-dependent actions.

7.3.1 Description of the Visualization. The grids in Figure 5 show the activity of the memory units during the interactions. Each row in the grids represents a time step in the interaction, starting from the top and ending at the bottom. The vertical axis labels show the shopkeeper and customer's behaviors that are input to the neural network at each time step, s_{t-1} and c_t . Additionally, the network's shopkeeper action predictions, s_t , are also shown. Note that sometimes the predicted s_t does not match the input s_{t-1} in the next time step, because the offline action predictions do not always match the simulated shopkeeper actions, which were used as input to the neural network.

Each column of Figure 5 represents a memory unit from Table 5. Each grid cell displays the state of the unit's gates with colored arrows—white (*preserve* memory), gray (*augment* memory), or black (*overwrite* memory). The remaining part of the cell represents the unit's output activation h_t —either -1 (blue) or 1 (red). Since a GRU's output activation is determined by a hyperbolic tangent function ($\tanh$), the activations tend to be pushed to one of the extremes of its range.

7.3.2 Remembering a Memory-setting Action. The rows at $t = 2$ are highlighted in yellow to indicate the occurrence of memory-setting actions. In the top grid, at this time step the customer states, "I'm looking for an expert camera." At this point, the GRU layer should encode a memory of "expert user" so the robot can subsequently perform appropriate memory-dependent actions. Note the activity of GRU unit 601 "expert user." At $t = 2$ the gates of unit 601 "expert user" are in the *overwrite* memory state, meaning that the previous GRU state, $h_{601,t=1} = 0$, is overwritten with information from the current input to the GRU layer, $x_{601,t=2}$. This causes the current GRU output, $h_{601,t=2}$, to change to 1. Since the GRU layer input, $x_{601,t=2}$, contains a condensed representation of the current network input, s_1, c_2 , it can be inferred that $h_{601,t} = 1$ indicates a memory of the occurrence of these input actions. The bottom grid shows a similar interaction, but where the relevant memory is that the customer wishes to take pictures of "sports."

The default activation of a memory encoding GRU unit may be 0, 1, or -1. In the majority of the GRU units visualized in Figure 5 the default activations are 0 (represented by white grid cells), but for GRU units 152 "pets" and 363 "sports" the default activations are 1 (red) and -1 (blue), respectively. Note that the activation for the "sports" unit in the second interaction changes to 1 when the corresponding memory-setting action occurs. Since there is no occurrence of a "pets" memory-setting action, the "pets" unit activation remains 1 throughout the interactions. Since the weights connecting the GRU layer's outputs to the final, output layer of the network (which decides the robot's next action) may be either positive or negative, there is no constraint on the values with which a GRU encodes memory.

7.3.3 Inferring the Meaning of a Memory Unit. The network inputs that have been stored in memory during any time step can be identified from a visualization like Figure 5. To determine the contents of a memory unit at time t , trace back through the memory unit's previous activations until the most recent memory *overwrite*, taking note of any memory *augmentations* on the way. The activation of the memory unit at time t represents the occurrence of customer or shopkeeper actions input at the times of the *overwrite* and *augmentations*. In this way, it may be possible in the future to display a GRU's memories in a readily human-readable way.

7.4 Distributed Representation of Memory

Neural networks generally represent information in a distributed manner. The visualizations of GRU activations during the two interactions in Figure 5 demonstrate an ideal case, where a single

Fig. 6. Heat map showing mutual information (MI) between memory labels and GRUs units. Units with high MI are important for representing memories. Memories have a distributed representation.

Fig. 7. Scatter plot showing mutual information (MI) between each GRU unit and each memory label. Points have been sorted by MI to better show the distribution of MI over the GRUs. Only a few GRUs have high MI and most have low MI, showing that some GRUs are specialized for representing the occurrence of memory-setting actions.

GRU is sufficient to represent the occurrence of a memory-setting action (e.g., “I’m looking for a camera to take pictures of *sports*.”). Further analysis reveals that the memories of such occurrences are distributed over many GRUs.

Figure 6 shows the mutual information (MI) between each memory type and each GRU unit. The average MI was 0.016 bits (std. dev. 0.009), the maximum was 0.116 bits, and the minimum was 0.0. This means that on average, 63.5 GRUs would be required to encode one memory type (since ideally a single bit could be used to encode whether or not the memory-setting action occurred). In fact, the distribution of MI over the GRUs is non-uniform, with most GRUs having very low MI and very few with high MI. This distribution can be seen more clearly in Figure 7.

8 DISCUSSION

8.1 The Effectiveness of the Proposed System in Learning a Memory Model

This study found that a robot was successfully able to learn a memory model from a large dataset of simulated example interactions that enabled it to correctly perform actions based on memories.

From the analysis in Section 7 it can be concluded that memories were encoded in the GRU layer of the robot’s neural network. That is, some GRU units encoded long-term memories about the customer’s actions. These GRU units work like a binary switch: when “on,” the unit represents that a specific memory-setting action occurred (e.g., the customer stated a preference for cameras well-suited for an expert photographer), and when “off,” it represents that the specific memory-setting action has not occurred. The final output layer of the neural network, which has connections leading from the GRU layer and decides the robot’s next action, uses the information stored in the GRU units to determine whether it is appropriate to perform a memory-dependent action, and which memory-dependent action is the most appropriate. The results of the offline evaluation, which showed that the system with a GRU layer for encoding memories significantly outperformed a system without a GRU layer, supports this conclusion.

In an ideal case, the number of GRU units that learn to encode memories should match the number of memory types that occurred in the training dataset (10). However, since the utterances in memory-setting actions have many natural variations of phrasing, it is challenging for the neural network to learn a perfect memory representation. For example, in the analysis in Section 7, only the units that were most critical for encoding the memories (based on mutual information with the ground truth memory labels) were examined, but the representation of memory was also distributed over some other GRU units with relatively high mutual information, as can be seen in Figures 6 and 7. This sort of distributed representation, which is common in neural networks, provides a challenge for future work to visualize memories during human-robot interaction at greater levels of detail.

8.2 Generalizability and Scalability

In the future, it would be interesting to apply the proposed system to other domains and real-world scenarios. In this work the proposed system was trained and tested on simulated interactions in a simplified scenario, and we believe it could successfully be applied to real-world scenarios. However, there are some limitations.

The proposed system was able to model several different memory types, which we expect would also occur in real-world camera shop interactions (Table 1). For example, we identified actions that depend on memories of the customer's level of knowledge, types of pictures the customer prefers to take, and camera features that the customer wants. In a general, real-world shopping scenario, we expect that the types of memories would be similar to what is found here; although, the number of customer types could be greater. This would necessitate a larger quantity of training data.

It is possible to estimate the required quantity of interaction data to train the proposed system. If there are m memory types, c customer types, s natural phrasings of each memory-setting action, d natural phrasings of each memory-dependent action, an average of i memory-dependent actions per interaction, and k examples are required by the machine learning algorithm to learn an accurate representation, then roughly $\frac{kmc s d}{i}$ training interactions would be required. In this work, we simulated interactions to get a sufficient number of training examples. To train the system for real-world scenarios that contain a greater variety of customer types and phrasings, a larger quantity of training data would be required. We believe that it will be possible to collect such datasets in the future with passive sensors set up in areas with high customer traffic.

One limitation of the proposed system is that it cannot generalize to memory types that are underrepresented in the training interactions. For example, the system cannot customize its behavior to idiosyncratic, infrequent customer types, such as *a customer seeking a gift for their father to use while scuba diving*. This is a limitation of data-driven HRI in general, since systems such as these are designed to learn only the repeatable, core behaviors of interaction. In the future it would be useful to detect idiosyncratic customer behaviors so the robot can take appropriate action. Furthermore, it may be necessary to combine the data-driven approach with other approaches, such as knowledge-based approaches, to deal with idiosyncratic situations.

9 CONCLUSION

In conclusion, this work attempts to answer the question of whether or not a shopkeeper robot is able to learn a memory model of human customer behavior to enable it to appropriately perform actions that depend on memory. To get a foothold in answering this question, a dataset of human-human interactions in a camera shop scenario was analyzed to discover the types of actions that involve memory. The analysis revealed several types of *memory-setting* and *memory-dependent* actions. Due to the wide diversity of natural utterance variations that occurred in the memory-setting and memory-dependent actions in the human-human dataset, and the variety of possible

interaction patterns, the number of examples of each interaction pattern was deemed too few to train a machine learning algorithm. Therefore, a dataset of simulated interactions was generated for that purpose. To train a shopkeeper robot, a learning system with a gated recurrent unit (GRU) recurrent neural network was proposed. The proposed system was compared to a baseline system without GRUs in an offline evaluation. A human evaluator found the actions of the proposed system to be correct at a significantly higher rate than the actions of the baseline. The trained neural network was analyzed with a visualization to reveal how it represented memories of customer behavior in its GRU layer. Finally, the results were discussed and some possibilities for future work were suggested.

10 APPENDIX – TABLES USED FOR INTERACTION SIMULATION

Table 7. Simulator Actions

Action	Actor
ENTERS	Customer
GREETES	Customer, Shopkeeper
WALKS_TO_CAMERA	Customer
LOOKING_FOR_A_CAMERA	Customer
LOOKING_FOR_A_CAMERA_WITH_X	Customer
EXAMINES_CAMERA	Customer
JUST_BROWSING	Customer
LEAVES	Customer
QUESTION_ABOUT_FEATURE	Customer
ASKS_FOR_SOMETHING_WITH_MORE_X	Customer
SILENT_OR_BACKCHANNEL	Customer
THANK_YOU	Customer, Shopkeeper
THINK_IT_OVER	Customer
DECIDE_TO_BUY	Customer
ANSWERS_QUESTION_X	Customer
NONE	Shopkeeper
ASKS_QUESTION_X	Shopkeeper
RETURNS_TO_COUNTER	Shopkeeper
LET_ME_KNOW_IF_YOU_NEED_HELP	Shopkeeper
ANYTHING_ELSE_I_CAN_HELP_WITH	Shopkeeper
ANSWERS_QUESTION_ABOUT_FEATURE	Shopkeeper
INTRODUCES_NEW_FEATURE	Shopkeeper
INTRODUCES_CAMERA	Shopkeeper
THIS_IS_MOST_X	Shopkeeper
APPENDED_PHRASE_RELATING_TO_MEMORY	Shopkeeper

Table 8. Picture Types and Preferred Features

Features	Feature is preferred for picture type									
	Art school	Everyday photos	Expert user	Family and friends	Nighttime	Novice user	Outdoors	Pets	Sports	Travel
Point and shoot	No	Yes	No	Yes	No	Yes	No	No	Yes	Yes
Mirrorless	No	No	No	No	No	No	No	No	No	No
DSLR	No	No	No	No	No	No	No	No	No	No
Colorful	No	No	No	No	No	No	No	No	No	No
Black	No	No	No	No	No	No	No	No	No	No
White	No	No	No	No	No	No	No	No	No	No
Silver	No	No	No	No	No	No	No	No	No	No
Purple	No	No	No	No	No	No	No	No	No	No
Pink	No	No	No	No	No	No	No	No	No	No
Red	No	No	No	No	No	No	No	No	No	No
Light	No	No	No	No	No	No	Yes	No	No	Yes
Heavy	No	No	No	No	No	No	No	No	No	No
Preset modes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes
Glamour retouch effects	No	Yes	No	Yes	No	Yes	No	Yes	No	No
Artistic modes	No	Yes	No	Yes	No	Yes	No	Yes	No	No
Cheap	Yes	No	No	No	No	Yes	No	No	No	No
Expensive	No	No	No	No	No	No	No	No	No	No
Autofocus points	No	Yes	No	Yes	No	No	No	Yes	Yes	No
Long exposure	Yes	No	Yes	No	Yes	No	Yes	No	No	No
Full manual control	Yes	No	Yes	No	No	No	Yes	No	Yes	No
Full frame	Yes	No	Yes	No	No	No	Yes	No	No	No
High ISO	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
Bulb mode	No	No	Yes	No	Yes	No	No	No	No	No

Table 9. Camera Features

Features	Camera has feature		
	Nikon	Sony	Canon
Point and shoot	Yes	No	No
Mirrorless	No	Yes	No
DSLR	No	No	Yes
Colorful	Yes	No	No
Black	Yes	Yes	Yes
White	No	Yes	No
Silver	Yes	Yes	No
Purple	Yes	No	No
Pink	Yes	No	No
Red	Yes	No	No
Light	Yes	Yes	No
Heavy	No	No	Yes
Preset modes	Yes	Yes	No
Glamour retouch effects	Yes	No	No
Artistic modes	No	Yes	No
Cheap	Yes	Yes	No
Expensive	No	No	Yes
Autofocus points	Yes	Yes	No
Long exposure	No	Yes	Yes
Full manual control	No	Yes	Yes
Full frame	No	No	Yes
High ISO	No	No	Yes
Bulb mode	No	No	Yes

REFERENCES

- [1] P. Liu, D. F. Glas, T. Kanda, and H. Ishiguro. 2016. Data-driven HRI: Learning social behaviors by example from human-human interaction. *IEEE Trans. Rob.* 32, 4 (2016), 988–1008.
- [2] P. Liu, D. F. Glas, T. Kanda, and H. Ishiguro. 2017. Learning proactive behavior for interactive social robots. *Auton. Rob.* 1–19.
- [3] P. Liu, D. F. Glas, T. Kanda, and H. Ishiguro. 2017. Two demonstrators are better than one—A social robot that learns to imitate people with different interaction styles. *IEEE Trans. Cog. Dev. Syst.* 2017. Early access.
- [4] P. Liu, D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. 2014. How to train your robot-teaching service robots to reproduce human social behavior. In *Proceedings of the RO-MAN'14*. IEEE, 961–968.
- [5] C. Olah. 2015. *Understanding LSTM Networks*. Colah's blog, colah.github.io/posts/2015-08-Understanding-LSTMs/.
- [6] K. Kuwamura, S. Nishio, and H. Ishiguro. 2016. Designing robots for positive communication with senior citizens. In *Proceedings of the IAS'13*. Springer, 955–964.
- [7] K. Kuwamura, S. Nishio, and S. Sato. 2016. Can we talk through a robot as if face-to-face? Long-term fieldwork using teleoperated robot for seniors with Alzheimer's disease. *Front. Psych.* 7 (2016).
- [8] S. Guo, J. Lenchner, J. H. Connell, M. Dholakia, and H. Muta. 2017. Conversational bootstrapping and other tricks of a concierge robot. In *Proceedings of the HRI'17*. 73–81.
- [9] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro. 2004. Interactive robots as social partners and peer tutors for children: A field trial. *Hum.-Comput. Interact.* 19, 1 (2004), 61–84.
- [10] M. Doering, D. F. Glas, and H. Ishiguro. 2019. Modeling interaction structure for robot imitation learning of human social behavior. *IEEE Trans. Hum.-Mach. Syst.* 49, 3 (2019), 219–231.
- [11] P. H. J. Kahn et al. 2010. Psychological intimacy with robots?: Using interaction patterns to uncover depth of relation. In *Proceedings of the HRI'10*. IEEE Press, 123–124.

- [12] H. A. Samani, A. D. Cheok, F. W. Ngai, A. Nagpal, and M. Qiu. 2010. Towards a formulation of love in human-robot interaction. In *Proceedings of the RO-MAN'10*. IEEE, 94–99.
- [13] H. Chen, X. Liu, D. Yin, and J. Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explor. Newslett.* 19, 2 (2017), 25–35.
- [14] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI'16*.
- [15] A. Sordoni et al. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the NAACL-HLT'15*.
- [16] Y. Kim, J. Bang, J. Choi, S. Ryu, S. Koo, and G. G. Lee. 2014. Acquisition and use of long-term memory for personalized dialog systems. In *Proceedings of the MA3HMI @ INTERSPEECH'14*. Springer, 78–87.
- [17] C. D. Kidd and C. Breazeal. 2008. Robots at home: Understanding long-term human-robot interaction. In *Proceedings of the IROS'08*. IEEE, 3230–3235.
- [18] R. Gockley et al. 2005. Designing robots for long-term social interaction. In *Proceedings of the IROS'08*. IEEE, 1338–1343.
- [19] I. Leite, C. Martinho, and A. Paiva. 2013. Social robots for long-term interaction: A survey. *Int. J. Soc. Rob.* 5, 2 (2013), 291–308.
- [20] A. M. Nuxoll and J. E. Laird. 2007. Extending cognitive architecture with episodic memory. In *Proceedings of the AAAI'07*. 1560–1564.
- [21] W. Duch, R. J. Oentaryo, and M. Pasquier. 2008. Cognitive architectures: Where do we go from here? In *Proceedings of the AGI'08*. 171 (2008) 122–136.
- [22] P. Baxter and G. Trafton. 2014. Cognitive architectures for human-robot interaction. In *Proceedings of the HRI'14*. IEEE, 504–505.
- [23] G. Fischer. 2001. User modeling in human–computer interaction. *User Mod. User-adapt. Interact.* 11, 1–2 (2001), 65–86.
- [24] W. Wahlster and A. Kobsa. 1989. User models in dialog systems. In *User Models in Dialog Systems*. Springer, 4–34.
- [25] A. Kobsa. 1990. User modeling in dialog systems: Potentials and hazards. *AI & Soc.* 4, 3 (1990), 214–231.
- [26] L. M. Hiatt, C. Narber, E. Bekele, S. S. Khemlani, and J. G. Trafton. 2017. Human modeling for human–robot collaboration. *Int. J. Rob. Res.* 36, 5–7 (2017), 580–596.
- [27] Z. C. Lipton, J. Berkowitz, and C. Elkan. 2015. A critical review of recurrent neural networks for sequence learning. Retrieved from: *Arxiv Preprint Arxiv:1506.00019*.
- [28] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.
- [29] K. Cho et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Retrieved from: *Arxiv Preprint Arxiv:1406.1078*.
- [30] S. Sukhbaatar, J. Weston, and R. Fergus. 2015. End-to-end memory networks. In *Proceedings of the NIPS'15*. 2440–2448.
- [31] A. Kumar et al. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the ICML'16*. 1378–1387.
- [32] A. Graves, G. Wayne, and I. Danihelka. 2014. Neural Turing machines. Retrieved from: *Arxiv Preprint Arxiv:1410.5401*.
- [33] H. Admoni and B. Scassellati. 2014. Data-driven model of nonverbal behavior for socially assistive human-robot interactions. In *Proceedings of the ICMIT'14*. ACM, 196–199.
- [34] I. Leite, A. Pereira, A. Funkhouser, B. Li, and J. F. Lehman. 2016. Semi-situated learning of verbal and nonverbal content for repeated human-robot interaction. In *Proceedings of the ICMIT'16*. ACM, 13–20.
- [35] J. Orkin and D. Roy. 2007. The Restaurant Game: Learning social behavior and language from thousands of players online. *J. Game Dev.* 3, 1 (2007), 39–60.
- [36] J. Orkin and D. Roy. 2009. Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of the AAMAS'09*. IFAAMAS, 385–392.
- [37] C. Breazeal, N. DePalma, J. Orkin, S. Chernova, and M. Jung. 2013. Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *J. Hum.-Rob. Interact.*, 2, 1 (2013), 82–111.
- [38] S. Chernova, N. DePalma, E. Morant, and C. Breazeal. 2011. Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In *Proceedings of the RO-MAN'11*. IEEE, 21–26.
- [39] S. Chernova, J. Orkin, and C. Breazeal. 2010. Crowdsourcing HRI through online multiplayer games. In *Proceedings of the AAAI Fall Symposium'10*. 14–19.
- [40] Z. Tian, R. Yan, L. Mou, Y. Song, Y. Feng, and D. Zhao. 2017. How to make context more useful? An empirical study on context-aware neural conversational models. In *Proceedings of the ACL'17 (Volume 2: Short Papers)*, 2 (2017), 231–236.
- [41] J. Perez, Y.-L. Boureau, and A. Bordes. 2017. Dialog system & technology challenge 6 overview of track 1-end-to-end goal-oriented dialog learning. In *Proceedings of the DSTC'17*.
- [42] L. E. Asri et al. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. Retrieved from: *ArXiv Preprint Arxiv:1704.00057*.

- [43] H. Schulz, J. Zumer, L. E. Asri, and S. Sharma. 2017. A frame tracking model for memory-enhanced dialogue systems. Retrieved from: *Arxiv Preprint Arxiv:1706.01690*.
- [44] S. Janarthanam and O. Lemon. 2014. Adaptive generation in dialogue systems using dynamic user modeling. *Comput. Ling.* 40, 4 (2014), 883–920.
- [45] D. Brscic, T. Kanda, T. Ikeda, and T. Miyashita. 2013. Person tracking in large public spaces using 3-D range sensors. *IEEE Trans. Hum.-Mach. Syst.* 43, 6 (2013), 522–534.
- [46] P. Langfelder, B. Zhang, and S. Horvath. 2008. Defining clusters from a hierarchical cluster tree: The dynamic tree cut package for R. *Bioinformatics* 24, 5 (2008), 719–720.
- [47] C. C. Aggarwal and C. Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*. Springer, 77–128.
- [48] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. Retrieved from: *ArXiv Preprint Arxiv:1412.6980*.
- [49] Y. Wu et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. Retrieved from: *Arxiv Preprint Arxiv:1609.08144*.
- [50] A. Vaswani et al. 2017. Attention is all you need. In *Proceedings of the NIPS’17*. 5998–6008.
- [51] J. Weston, S. Chopra, and A. Bordes. 2015. Memory networks. In *Proceedings of the ICRL’15*.
- [52] T. M. Cover and J. A. Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.
- [53] B. C. Ross. 2014. Mutual information between discrete and continuous data sets. *PLoS ONE*, 9, 2 (2014), e87357.
- [54] Z. Li et al. 2018. Towards binary-valued gates for robust LSTM training. In *Proceedings of the ICML’18*.
- [55] M. Doering, P. Liu, D. F. Glas, T. Kanda, D. Kulić, and H. Ishiguro. 2019. Curiosity did not kill the robot: A curiosity-based learning system for a shopkeeper robot. *ACM Trans. Hum.- Robot Interact.* 8, 3, Article 15 (July 2019), 24 pages. <https://doi.org/10.1145/3326462>

Received August 2018; revised March 2019; accepted May 2019