Learning End-to-End Goal-Oriented Dialog with Maximal User Task Success and Minimal Human Agent Use

Janarthanan Rajendran[†][*], **Jatin Ganhotra**[‡], and **Lazaros C. Polymenakos**[‡]
[†]University of Michigan
[‡]IBM Research
rjana@umich.edu, jatinganhotra@us.ibm.com, and lcpolyme@us.ibm.com

Abstract

Neural end-to-end goal-oriented dialog systems showed promise to reduce the workload of human agents for customer service, as well as reduce wait time for users. However, their inability to handle new user behavior at deployment has limited their usage in real world. In this work, we propose an end-to-end trainable method for neural goal-oriented dialog systems that handles new user behaviors at deployment by transferring the dialog to a human agent intelligently. The proposed method has three goals: 1) maximize user's task success by transferring to human agents, 2) minimize the load on the human agents by transferring to them only when it is essential, and 3) learn online from the human agent's responses to reduce human agents' load further. We evaluate our proposed method on a modified-bAbI dialog task,[1] which simulates the scenario of new user behaviors occurring at test time. Experimental results show that our proposed method is effective in achieving the desired goals.

1 Introduction

Neural end-to-end dialog systems showed huge potential for various goal-oriented dialog tasks such as restaurant reservation, flight ticket booking, and hotel reservation. However, their use in the real world has been limited because of their inability to handle new user behavior at deployment.

There are two main methods to build neural end-to-end goal-oriented dialog systems. In the first method, large amounts of human–human chat logs of a particular task are collected and then the dialog system is trained to mimic the chat logs using Supervised Learning (SL) (Bordes et al., 2017). In the second method, the dialog system is trained to complete the task against a human (user) simulator (Young et al., 2013). The training is done using Reinforcement Learning (RL) by providing reward for task completion and also intermediate rewards for pre-identified subtask completion. This is often accompanied by a SL pre-training, as in Liu and Lane (2017).

Dialog systems built using either of these methods would fail in the presence of new user behaviors during deployment, which were missing in training. Failure here refers to the inability of the dialog system to complete the task for the user. The new behaviors can be a different way of a user asking/providing certain information or could also be as simple as an user utterance with out-of-vocabulary words. The failure happens when these new behaviors are beyond the generalization capabilities of the trained systems due to the limited coverage of training data collected. For a real-world use-case, it is difficult to collect chat logs and/or to build a user simulator that covers all possible user behaviors, which implies that users with new behaviors are bound to come by during deployment. The new user behaviors that we refer to and focus here are only those that are valid and acceptable, that is, these user behaviors should ideally have been part of the training data (chat logs in the case of SL and user simulator behavior pattern in the case of RL).

For an enterprise that uses this dialog system, these failures could affect their business directly. In addition to losing customers who faced these failures, the enterprise also loses future users/customers, as it also affects the perceived reliability of the dialog system and, hence,

[*] This work was done when the author was an intern at IBM Research, NY.

[1] modified-bAbI dialog tasks - `https://github.com/IBM/modified-bAbI-dialog-tasks`.

Transactions of the Association for Computational Linguistics, vol. 7, pp. 375–386, 2019. Action Editor: James Henderson.
Submission batch: 11/2018; Revision batch: 2/2019; Published 7/2019.

the enterprise's reputation. Although the dialog system fails for new user behaviors, it can perform the task well for majority of user behaviors. However, these failures have restricted the deployment of neural end-to-end goal-oriented dialog systems and have forced the enterprises to rely completely on human agents or other systems.

There has been recent work on using using user feedback (Liu and Lane, 2017) and active human teaching (Liu et al., 2018) during deployment to improve robustness to new user behaviors. Although they have produced better systems, they are not good enough to be deployed in the real world. In this work, we propose a method that can be used in addition to the aforementioned ideas and enables a dialog system to perform well in the real world.

We propose an end-to-end trainable method, in which the dialog system can automatically identify a new user behavior during deployment that the system might fail at and transfer the task to a human agent, such that the user's task is completed without any issue. At the same time, the dialog system also learns from the human agent's response to handle that new user behavior in future. Our method also allows one to choose the trade-off between maximizing user's task success and minimizing the workload on human agents.

We set the following three goals for our method:

- Maximize task success rate for the user by transferring to a human agent in cases where the dialog system might fail.
- Minimize the use of human agent by transferring to the human agent only when it is essential.
- Learn online from the human agent's response to reduce the use of human agent over time.

This paper is organized as follows. Section 2 describes the proposed method. In Section 3, we introduce modified-bAbI dialog tasks, which simulate new user behavior occurrence at deployment and serve as testbed for evaluating our proposed method. Section 4 details our experimental results on modified-bAbI dialog tasks. Section 5 discusses related work and Section 6 concludes.

2 Proposed Method

Consider a neural dialog model M trained for a goal-oriented dialog task. We also have a human agent H who is trained for the same task. Both M and H can take the dialog so far as input and produce the response for the user utterance (u). There is a neural classifier C that uses the dialog state vector (s) from model M as input and decides whether to use the model M to provide response to the user or to transfer to the human agent H who could then provide the response to the user. The proposed method is shown in Figure 1 (right).

In a real-world setting, we cannot expect the same exact user utterances to come during deployment that the model came across during its training. Therefore, for a new dialog, it is not possible to know beforehand whether model (M) would provide a correct response or not. The classifier (C) has to learn this through trial and error and generalize. Therefore, the classifier is trained using RL.

The classifier is provided a high reward if it chooses the model M and the model produces a correct/valid response. The classifier is provided a relatively smaller reward if it chooses the human agent instead. We assume that the human agent's response is always correct. If the classifier chooses the model M and the model provides an incorrect response, the classifier (C) is penalized heavily. The validation, if a response is correct or not, is provided by the user as feedback. The classifier is trained using RL to make decisions (take actions) in order to maximize this reward function. The reward function helps achieve two of our aforementioned goals:

- *Maximize task success rate for the user*: The reward function encourages the classifier to learn the dialog scenarios in which the model M might fail and choose a human agent instead. Therefore, the classifier helps to avoid sending an incorrect response to the user from the model.
- *Minimize human agent use*: The reward function also encourages the classifier to learn, identify, and choose the model M in cases where the model has a high chance of providing the correct response, as the classifier gets a higher reward compared to choosing a human agent. This minimizes the use of human agent to only when it is essential.

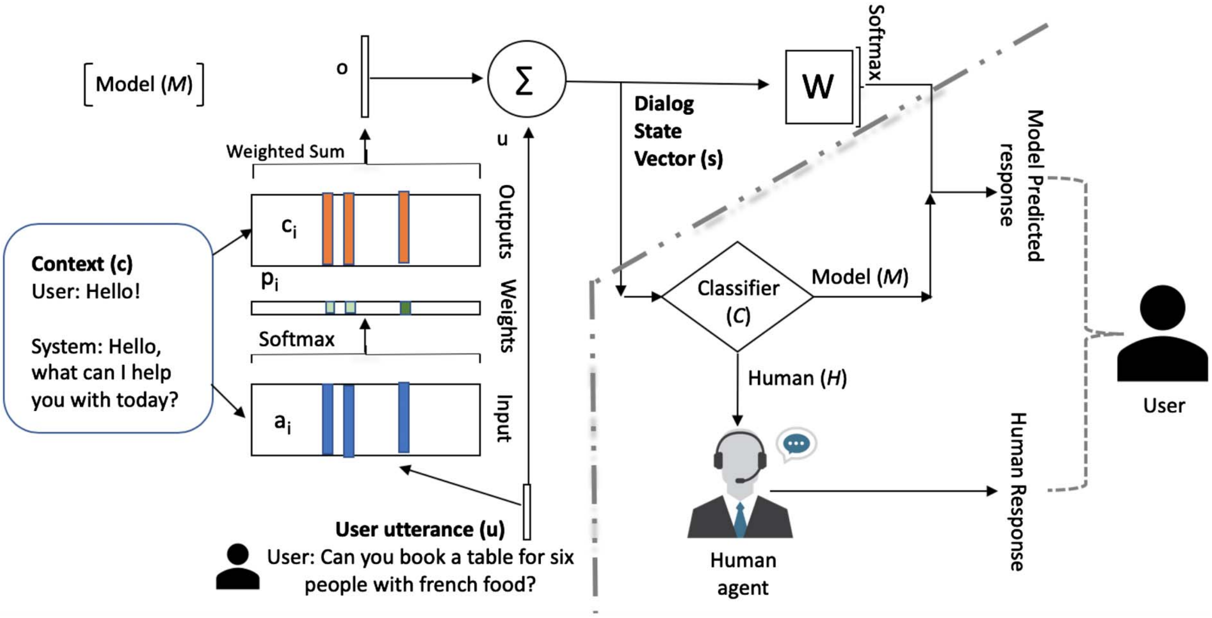


Figure 1: ***Left***: A single layer version of memN2N (model M). ***Right***: Proposed method.

Here is an example reward function, which would achieve the desired goals:

- $+1$: if human H is chosen
- $+2$: if model M is chosen and the model's response is correct
- -4: if model M is chosen and the model's response is incorrect

The reward function allows the designer to choose the trade-off between maximizing the user task completion versus minimizing the human agent's workload. For example, when the model (M) is chosen, increasing the positive reward if the model's response is correct and reducing the penalty when the model's response is incorrect would encourage the overall system to use model (M) more to respond to the user.

The gradient updates obtained for the classifier through the rewards received are also propagated back to the model M through the dialog state vector. This trains the model M to incorporate a) essential information about the dialog so far and b) the model's confidence in producing the correct response, in the dialog state vector, such that the classifier can utilize it to make the right decision.

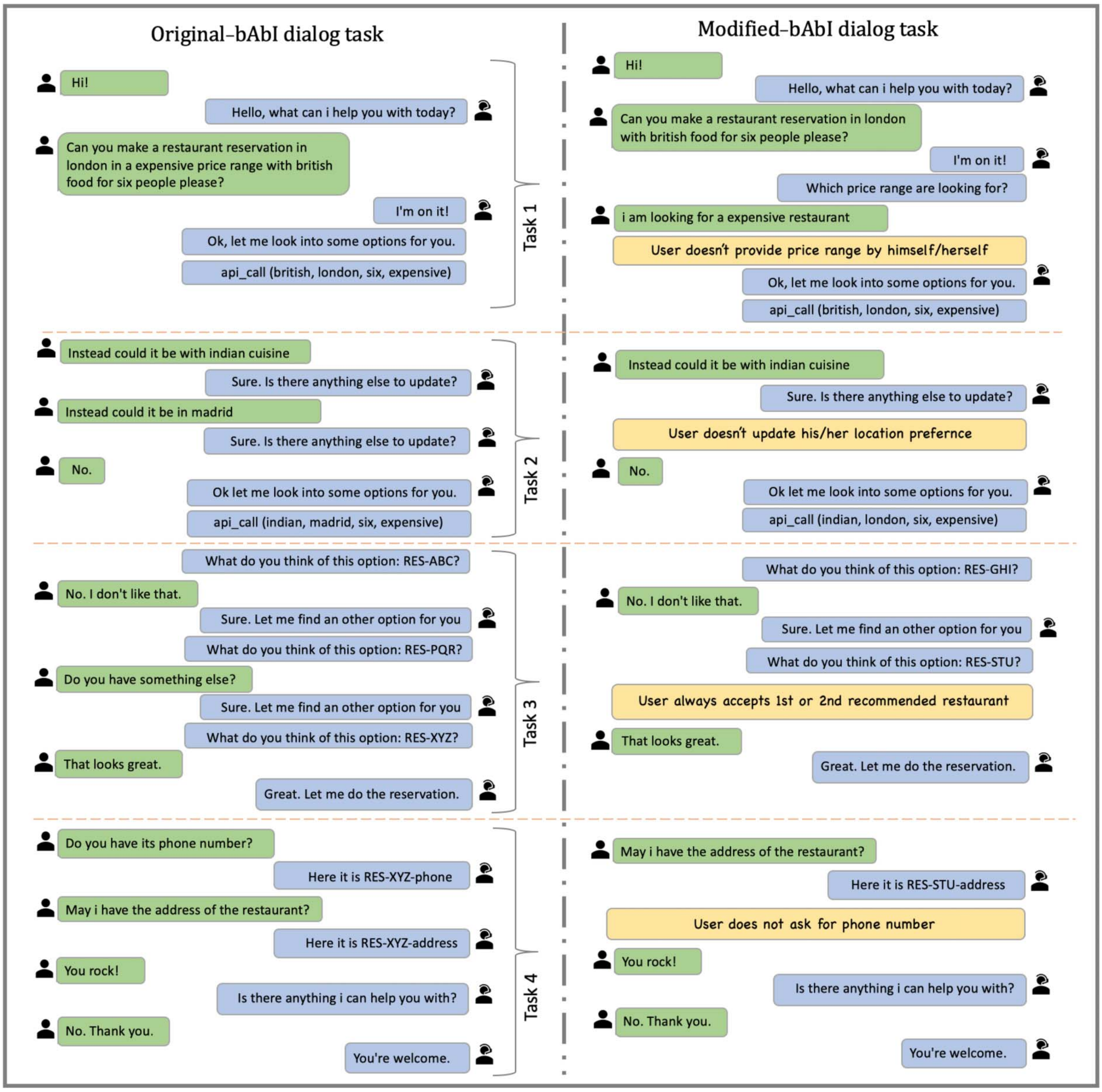
Whenever the classifier chooses the human agent (H), the dialog interaction (including the human response) is also added to the training data of the model (M) and the model is updated online using SL. This helps achieve our third goal:

- *Reduce human agent use over time*: The online update allows the model M to respond to the corresponding new user behavior and provide the correct response if a similar dialog scenario occurs in the future. This also enables the classifier to reduce its dependence on the human agent (H) over time.

The classifier keeps changing during its lifetime to adapt to the changes in the model M. Note that a human agent is involved only when the classifier transfers the dialog to a human. The idea is generic enough to be used with any neural dialog model (M)—for example, HRED (Sordoni et al., 2015) or end-to-end memory network (Sukhbaatar et al., 2015). In our experiments, we use the end-to-end memory network for our model M and a multi-layer perceptron (MLP) for the classifier C. Additional details on the model architecture and training are provided in Section 4.

3 Modified bAbI dialog tasks

bAbI dialog tasks (referred to as original-bAbI dialog tasks from here on) were proposed by Bordes et al. (2017) as a testbed to evaluate the strengths and shortcomings of end-to-end dialog systems in goal-oriented applications (Seo et al., 2017; Rajendran et al., 2018). The data set is generated by a restaurant reservation simulation where the



Original-bAbI dialog task

Hi!

Hello, what can i help you with today?

Can you make a restaurant reservation in london in a expensive price range with british food for six people please?

I'm on it!

Ok, let me look into some options for you.

api_call (british, london, six, expensive)

Instead could it be with indian cuisine

Sure. Is there anything else to update?

Instead could it be in madrid

Sure. Is there anything else to update?

No.

Ok let me look into some options for you.

api_call (indian, madrid, six, expensive)

What do you think of this option: RES-ABC?

No. I don't like that.

Sure. Let me find an other option for you

What do you think of this option: RES-PQR?

Do you have something else?

Sure. Let me find an other option for you

What do you think of this option: RES-XYZ?

That looks great.

Great. Let me do the reservation.

Do you have its phone number?

Here it is RES-XYZ-phone

May i have the address of the restaurant?

Here it is RES-XYZ-address

You rock!

Is there anything i can help you with?

No. Thank you.

You're welcome.

Modified-bAbI dialog task

Hi!

Hello, what can i help you with today?

Can you make a restaurant reservation in london with british food for six people please?

I'm on it!

Which price range are looking for?

i am looking for a expensive restaurant

User doesn't provide price range by himself/herself

Ok, let me look into some options for you.

api_call (british, london, six, expensive)

Instead could it be with indian cuisine

Sure. Is there anything else to update?

User doesn't update his/her location prefernce

No.

Ok let me look into some options for you.

api_call (indian, london, six, expensive)

What do you think of this option: RES-GHI?

No. I don't like that.

Sure. Let me find an other option for you

What do you think of this option: RES-STU?

User always accepts 1st or 2nd recommended restaurant

That looks great.

Great. Let me do the reservation.

May i have the address of the restaurant?

Here it is RES-STU-address

User does not ask for phone number

You rock!

Is there anything i can help you with?

No. Thank you.

You're welcome.

Figure 2: **Modified-bAbI dialog task**. A user (in green) chats with a dialog system (in blue) to book a table at a restaurant. We update each subtask in the bAbI dialog task with specific changes (in yellow).

final goal is booking a table. The simulator uses a knowledgebase (KB) which contains information about restaurants. There are five tasks: Task 1 (Issuing API calls), Task 2 (Updating API calls), Task 3 (Displaying options), Task 4 (Providing extra information), and Task 5 (Conducting full dialogs). Tasks 1 and 2 test the dialog system to implicitly track dialog state, Tasks 3 and 4 check if the system can learn to use the information from KB, and Task 5 combines all tasks. It is a retrieval task, where the dialog system has to retrieve the correct response from a set of given candidate responses.

We propose modified-bAbI dialog tasks, an extension of original-bAbI dialog tasks (Bordes et al., 2017). We modify the original-bAbI dialog tasks by removing and replacing certain user behaviors from the training and validation data. The test set is left untouched. This simulates a scenario where some new user behaviors arise during the test (deployment) time that were not seen during the training and hence allows us to test our proposed method. This also mimics real-world data collection via crowdsourcing in the sense that certain user behavior is missing from the training data. Figure 2 shows a dialog sample

Behavior type	# dialogs
Task-1	494
Task-2	539
Task-3	561
Task-4	752

Table 1: Modified-bAbI test set statistics. The numbers shown represent number of dialogs in test data (Task 5) out of 1,000 dialogs where a certain type of new user behavior is encountered.

from modified-bAbI dialog tasks.[2] We propose the following changes to the training and validation data of the original-bAbI dialog tasks:

- In Task 1, a user places a request that can contain from 0 to 4 of the required fields to book a table. The system asks questions to retrieve values for the missing fields and generates the correct corresponding API call. In modified-bAbI dialog tasks, the user doesn't provide the value for the price range by themselves and only provides that information when asked by the system.

- In Task 2, the user can request the system to update any of their preferences (cuisine, location, price range, and number of people). In modified-bAbI dialog tasks, the user doesn't update their location preference.

- In Task 3, for the API call matching the user request, information retrieved from the KB is provided as part of dialog history. The system must propose options to the user by listing the restaurant names sorted by their corresponding rating (in decreasing order). The system keeps proposing a new restaurant until the user accepts. In modified-bAbI dialog tasks, the user always accepts first or second recommended restaurant.

- In Task 4, the user can ask for the phone number or address for the selected restaurant. In modified-bAbI dialog tasks, the user does not ask for the phone number.

We incorporate this changes to the final Task 5 (Conducting full dialogs). We perform experiments on modified-bAbI dialog task-5, as Tasks 1−4 are subsets of a full conversation and don't represent a complete meaningful real-world conversation standalone. The statistics for new user behavior in the test set (which was left untouched) are shown in Tables 1 and 2.

# new behavior	0	1	2	3	4
# dialogs	20	178	350	340	112

Table 2: Modified-bAbI test set statistics. The numbers shown represent number of dialogs in test data (Task 5) out of 1,000 dialogs, where no new user behavior or one or more type of new user behavior is encountered.

4 Experiments and Results

4.1 Baseline method

A dialog model M is trained on the modified dialog bAbI task and is used for deployment. The model is not updated during test/deployment. An end-to-end memory network (Sukhbaatar et al., 2015) architecture is used for the model M. End-to-end memory networks (memN2N) are an extension of memory networks proposed by Weston et al. (2014). They have been successful on various natural language processing tasks and perform well on original-bAbI dialog tasks. Hence, we chose them for our model M.

A single-layer version of the memN2N model is shown in Figure1 (left). A given sentence (i) from the context (dialog history) is stored in the memory by: a) its input representation (a_i) and b) its output representation (c_i), where each memory contains the embedding representation for that sentence. The embedding representation of the sentence is calculated by adding the embeddings of all the words in the sentence (Bordes et al., 2017). Attention of current user utterance (u) over memory is computed via dot product, to identify the relevance of the memory with respect to the current user utterance (query). (p_i) represents the probability for each memory based on attention scores (equation 1). An output vector (o) is computed by the weighted sum of memory embeddings (c_i) with their corresponding probabilities (p_i) (equation 2). The output vector (o) is the overall context embedding and, with query (u) represents the dialog state vector (s) (equation 3). The attention can be done multiple times, namely, multiple layers (3 in our experiments), by updating u with s and repeating equations (1), (2), and (3). The last internal state

[2] modified-bAbI dialog tasks - https://github.com/IBM/modified-bAbI-dialog-tasks.

Data Set	Per-turn	Per-dialog
Original-bAbI	98.5	77.1
Modified-bAbI	81.7	3.7

Table 3: Test results (accuracy %) for our baseline method (M:memN2N) across original and modified bAbI dialog tasks.

is then used as the dialog state vector s to select the candidate response for the model M and also provided as input to the classifier C.

$$p_i = \text{Softmax}(u^T(a_i)) \tag{1}$$

$$o = \sum_i p_i c_i \tag{2}$$

$$s = (o + u) \tag{3}$$

The model is trained using SL on the training data. The trained model is then used during deployment. In our case, deployment is same as testing the model on the test data. Our results for the baseline method for the original and modified bAbI dialog tasks are given in Table 3. The hyperparameters used for training the model M in the baseline method are provided in Appendix A.1.

Per-turn accuracy measures the percentage of responses that are correct (i.e., the correct candidate is selected from all possible candidates). Note that, as mentioned in Section 3, we only modify the training and validation sets, and use the same test set. Per-dialog accuracy measures the percentage of dialogs where every response is correct. Therefore, even if only one response in a dialog is incorrect, this would result in a failed dialog (i.e., failure to achieve the goal of booking a table).

From Table 3, we observe that the baseline method of using the trained memN2N model performs poorly on modified-bAbI dialog tasks, which has new user behaviors at test time that the model has not seen during training. For modified-bAbI dialog tasks, the baseline method achieves 81.7% on per-turn accuracy and the per-dialog accuracy decreases to 3.7%. This implies that the majority of dialogs would be unsuccessful in completing the goal. These results clearly demonstrate that the baseline method (end-to-end memory network) does not perform well on our proposed testbed, which simulates new user behaviors occurrence at deployment.

4.2 Proposed Method: $M^* + C^*$

We use the same memN2N model used in the baseline method for the model M here. However, in our proposed method, we also have a classifier C that takes as input the dialog state vector s and makes a decision on whether to use the model to respond to the user or to transfer the dialog to a human agent. For our experiments, C is a MLP that outputs a probability distribution over the two actions. The sampled action is performed and the user feedback is used to determine the reward obtained, which is then used to train the classifier and the model.

The following scenarios arise during deployment depending on the classifier's action and the model's prediction:

- The classifier could choose a human agent H to respond to the user. Because we use the test data of the modified-bAbI dialog task as a way to test deployment performance, we already have the ground truth labels for the different dialog scenarios that arise during the deployment. We use these ground truth labels as the human response.
- The classifier chooses the model M and the model produces a correct or incorrect response. In real world, this validation/ feedback on whether the response was correct or not is obtained from the user during deployment. For our experiments, we use the ground truth labels for test data to provide the validation from the user. In a sense, we mimic an ideal user using the test data ground truth labels.

We have two versions of the proposed method: a) the model trained on the training data is kept fixed during deployment and only the classifier is updated ($M + C^*$), and b) both the model and the classifier are updated ($M^* + C^*$) during deployment. The classifier is trained from scratch using REINFORCE (Williams, 1992). For $M^* + C^*$, the model is updated using the following three ways:

- The gradients obtained for the classifier C are passed through the dialog state vector s to the model.
- The human responses provided for cases where the classifier transferred the dialog

Method		User Accuracy		Model ratio	Final Model Accuracy	
		Per-turn	Per-dialog		Per-turn	Per-dialog
Baseline method (M)		81.73(0)	3.7(0)	100.0(0)	81.73(0)	3.7(0)
R: 1, 2, -4	$M + C^*$	92.85(1.58)	33.48(10.59)	51.97(8.22)	81.73(0)	3.7(0)
	$M^* + C^*$	96.28(1.16)	54.5(10.72)	64.06(4.65)	90.83(0.82)	14.82(3.7)
	$M_a^* + C^*$	96.19 (1.21)	54.44(11.40)	61.14(6.9)	88.98(0.34)	10.26(1.39)
R: 1, 3, -3	$M + C^*$	91.31(1.15)	26.50(7.57)	58.82(4.62)	81.73(0)	3.7(0)
	$M^* + C^*$	94.67(1.20)	43.48(8.80)	70.33(2.13)	89.27(0.74)	12.84(2.22)
	$M_a^* + C^*$	94.08(1.0)	38.8(8.15)	69.69(6.14)	88.75(0.91)	11.62(2.61)

Table 4: Test results for the different methods on the modified-bAbI dialog task. The numbers represent the mean and standard deviation (shown in parenthesis) of running the different methods across 5 different permutations of the test set. *User Accuracy*: Task success rate for the user; *Model ratio*: Percentage of time the classifier chooses the model M; *Final Model Accuracy*: Accuracy of the model M at the end of testing.

to a human agent, are added to the training data to augment it and are also used to update the model using supervised learning.

- Dialogs are sampled from the augmented training data and are used to update the model using supervised learning to avoid forgetting.

The $M^* + C^*$ method uses a fixed number of samples (e.g., 2 batches for our experiments) from augmented training data to update the model. We also implement and evaluate a variant of $M^* + C^*$ method: $M_a^* + C^*$ where the number of samples are decided based on the model M's performance on validation data. During deployment, after each batch of test data, the model is evaluated on the validation data. The difference between the current validation per-turn accuracy ($v_{acc}^{current}$) and the best validation per-turn accuracy so far (v_{acc}^{best}) estimates the loss in information learned from original training data during deployment. This is used to determine the number of batches ($b \geq 0$) for updating the model, as per the equation:

$$b = \alpha * (v_{acc}^{current} - v_{acc}^{best}) \tag{4}$$

The $M_a^* + C^*$ (a = adaptive) method tries to update the model with the right number of dialog samples from the augmented data. Further details on training are provided in Appendix A.2. We perform our experiments on two sets of reward functions. The first reward function is as follows:

- $+1$: if human H is chosen
- $+2$: if model M is chosen and the model's response is correct
- -4: if model M is chosen and the model's response is incorrect

The results are shown in Table 4. The test data are provided sequentially, which mimics the deployment in a real-world setting. Because the performance depends on the order in which the test dialogs are provided, we evaluate our proposed method on 5 different permutations of the test set. We present the mean and standard deviation of the performance measures across the 5 permutations. The results for the individual permutations are provided in Appendix B. The performance measures used for evaluation are:

- *User Accuracy*: Task success rate as perceived by the user, irrespective of whether the response is provided by the human agent H or the model M. This measures our goal #1 - Maximize task success rate.
- *Model ratio*: Percentage of time the classifier C provides the model response to the user (i.e., human agent is not involved). This measures our goal #2 - Minimize human agent use.
- *Final Model Accuracy*: Accuracy of the model M on the test data at the end of testing. This is obtained by evaluating the model M on the test data again after the testing phase is over. This measures our goal #3 - Reduce human agent use over time, by online learning of the model.

While the baseline method M gets a per-turn user accuracy of 81.73%, using and learning a classifier ($M + C^*$) achieves user accuracy of 92.85%, an increase of more than 10 percentage points. If the model is also updated during the deployment ($M^* + C^*$), we observe further increase in per-turn accuracy (96.28%). Whereas $M + C^*$ achieves better performance by using the model 51.97% of the time, $M^* + C^*$ achieves high accuracy by using the model even more (64.06%), thereby reducing the human agent's workload. This is attributed to the improvement in the model during the deployment for the $M^* + C^*$ method. This is supported by the improvement in the model accuracy, going from 81.73% at the start of test to 90.83% by the end (shown as the final model M's accuracy in Table 2). We observe that $M^*_a + C^*$ does not provide an improvement, but performs similarly to $M^* + C^*$ on all performance measures. The numbers reported are calculated as the running average of the different performance measures by evaluating on the fixed size test data (1,000 dialogs) once sequentially. We expect an improvement in the various performance measures over longer periods of deployment (test time).

The reward function determines the trade-off between the user's task success rate and the human agent's load. We perform additional experiments by modifying the reward function to showcase this trade-off. For example, if we want to reduce the load on the human agent further, we can increase the reward provided when the model M is chosen and the model's response is correct and decrease the penalty when the model is chosen and model's response is incorrect. One such reward function is as follows:

- +1: if human H is chosen
- +3: if model M is chosen and the model's response is correct
- -3: if model M is chosen and the model's response is incorrect

The results for the new reward function are shown in the last two rows of Table 4. In comparison with performance measures for reward function (1,2,-4), for both methods—$M + C^*$ and $M^* + C^*$—we observe a small drop in the user accuracy and a significant increase in model ratio, which showcases our intended goal in altering the reward function.

5 Related Work

Most of the successful goal-oriented dialog learning systems in the past have been based on slot-filling for domain-specific tasks (Schatzmann et al., 2006; Singh et al., 2000). These include Markov decision process (MDP) based (Levin et al., 2000; Pieraccini et al., 2009) and partially observable Markov decision process (POMDP) based (Young et al., 2013; Gasic et al., 2013) systems. These are RL systems that model and track state transitions and take appropriate actions (dialog utterances) to obtain information from the user to fill the desired slots. They require hand-crafted features for state and action space representations and hence are restricted to very narrow settings.

Recently there has been a great deal of interest in building end-to-end neural dialog systems for goal-oriented dialog tasks. Both supervised learning-based (training the model on collected chat logs of the dialog task) (Bordes et al., 2017; Eric and Manning, 2017; Wen et al., 2017) and deep RL-based systems (Zhao and Eskenazi, 2016; Li et al., 2017b; Peng et al., 2017) have been studied. For RL systems, training the model from scratch requires many interactions. Hence, RL systems are often augmented with SL-based pre-training on collected chat logs of the dialog task (Henderson et al., 2008; Williams et al., 2017; Liu and Lane, 2017).

Training models through RL by using user feedback during deployment makes the system more robust to new user behaviors (Williams and Zweig, 2016; Liu and Lane, 2017). There has also been recent work on actively using human in the loop to teach and assist the learning of neural dialog systems (Li et al., 2017a; Liu et al., 2018).

Although these approaches have focused on different ways to improve the neural goal-oriented dialog systems and maximize user success rate by a) improving the model, b) implementing better ways of online learning, or c) using human teaching; the problem of handling new user behaviors during deployment has not yet been solved. Our proposed method directly optimizes for maximum user success and provides a framework where existing techniques for model learning, online learning, and human teaching can be used in tandem, to enable the end-to-end goal-oriented dialog systems ready for real-world use.

6 Conclusion and Future Work

Our proposed method provides a new framework for learning and training goal-oriented dialog systems for real-world use. The proposed method allows us to maximize user success rate by minimally using human agents instead of the dialog model for cases where the model might fail. Our evaluation on the modified-bAbI dialog task shows that our proposed method is effective in achieving the desired goals.

We introduce a new method for designing and optimizing goal-oriented dialog systems geared for real-world use. Our method allows the designer to determine the trade-off between the desired user's task success and human agent workload. We believe this opens up a new and promising research direction that would spark an increase in the use of end-to-end goal-oriented dialog systems in the real world soon.

There are several limitations to our current evaluation, which we discuss here and hope to overcome in future work. We use simple techniques for the different components in our method, but these can be replaced with more sophisticated state-of-the-art techniques for improved performance in terms of absolute values. For example, whereas we use REINFORCE, an on-policy method for training the classifier, it would be interesting to try off-policy reinforcement learning techniques to use the samples more effectively. We could also try state-of-the-art online learning methods to see how they affect the performance.

In our experiments, the learning of the classifier C starts from scratch during the deployment. In our future work, we are interested in exploring ways of pre-training the classifier C before deployment, so that the learning of C can happen faster, with less samples during deployment. We are also interested in drawing ideas from novelty detection methods to see if they can help the classifier C to generalize better.

Note that, for our experiments, we use an artificially constructed data set—modified bAbI dialog tasks—which incorporates two essential assumptions: a) a perfect human agent and b) correct user feedback. For actual real-world deployments with real users, although the former assumption might still hold true, the latter might not always be true. In our future work, we are interested in relaxing these assumptions and evaluating our proposed method on actual real-world deployments with real users.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *In the International Conference on Learning Representations (ICLR)*.

Mihail Eric and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 214–222.

James Henderson, Oliver Lemon, and Kallirroi Georgila 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017a. Dialogue learning with human-in-the-loop. *In the International Conference on Learning Representations (ICLR)*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017b. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP)*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? Research in commercial spoken dialog systems. In *Proceedings of the 12th International Conference on Text, Speech and Dialogue*, TSD '09, pages 3–13.

Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh, and Lazaros Polymenakos. 2018. Learning end-to-end goal-oriented dialog with multiple answers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(2):97–126.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Satinder Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A. Walker. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 645–651.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jason D. Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *CoRR*, abs/1606.01269.

Ronald J. Williams 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4): 229–256.

Steve J. Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Method	User Accuracy		Model ratio	Final Model Accuracy	
	Per-turn	Per-dialog		Per-turn	Per-dialog
Baseline method (M)	81.73	3.7	100.0	81.73	3.7
Reward: 1, 2, -4 ($M^* + C^*$)	97.49	65.2	62.15	91.33	18.2
	96.4	55.1	60.45	90.58	12.7
	94.47	38.3	71.13	89.54	9.7
	97.07	62.9	60.23	91.65	18.4
	95.99	51.0	66.35	91.06	15.1
Reward: 1, 3, -3 ($M^* + C^*$)	92.77	29.2	73.7	88.52	11.4
	94.23	41.3	68.6	88.79	12.3
	95.22	46.5	70.1	89.19	11.3
	95.47	49.2	68.45	89.39	12.5
	95.68	51.2	70.8	90.46	16.7

Table 5: Test results for $M^* + C^*$ method on different permutation of modified-bAbI dialog task's test set

Method	User Accuracy		Model ratio	Final Model Accuracy	
	Per-turn	Per-dialog		Per-turn	Per-dialog
Baseline method (M)	81.73	3.7	100.0	81.73	3.7
Reward: 1, 2, -4 ($M_a^* + C^*$)	96.15	51.8	64.64	88.52	9.9
	94.69	43.4	66.96	89.16	8.8
	96.52	56.5	56.57	88.75	9.6
	97.99	72.9	51.26	89.37	12.5
	95.63	47.6	66.27	89.12	10.5
Reward: 1, 3, -3 ($M_a^* + C^*$)	92.51	26.9	79.16	87.81	9.9
	95.23	49.3	65.67	88.8	12.0
	94.03	36.4	70.06	87.82	8.2
	94.55	41.4	63.03	89.73	14.8
	94.1	40.0	70.57	89.57	13.2

Table 6: Test results for $M_a^* + C^*$ method on different permutation of modified-bAbI dialog task's test set.

A. Appendix: Training Details

A.1 Baseline Method: (M)

The hyperparameters used for the training the memN2N model in our baseline method are as follows: hops = 3, embedding_size = 20, batch_size = 32. The entire model is trained using stochastic gradient descent with learning rate = 0.01 and annealing (anneal_ratio = 0.5, anneal_period = 25), by minimizing the standard cross-entropy loss between the predicted response and the correct response. We learn two embedding matrices A and C for encoding context (input and output representations) and a separate embedding matrix B for encoding the query. We use position encoding for encoding word position in the sentence (Sukhbaatar et al., 2015). We also add temporal features to encode information about the speaker for the given utterance (user/system), similar to Bordes et al. (2017) and weight matrices TA and TC are learned for encoding temporal features. The same weight matrices are reused for the 3 hops. We used 599 as the random seed for both tf.set_random_seed and tf.random_normal_initializer for our embedding matrices. The test results reported for the baseline method are calculated by choosing the model with the highest validation per-turn accuracy across multiple runs.

A.2 Proposed Method: ($M^* + C^*$)

We use the same hyperparameters as the baseline method of A.1 for training the model M.

The classifier C is trained using REINFORCE (Williams, 1992) with a learning rate of 0.01. In $M + C^*$, after every batch of test data, the classifier MLP is updated. In $M^* + C^*$, after every batch of the test data (deployment), along with the classifier MLP update, the model is also updated in the three ways discussed. In $M^* + C^*$ the update using the human responses is done multiple times after every batch (3 in our case). For the update with the training data, two batches of training data are randomly sampled after every batch of test data.

B. Appendix: Extended Results

Tables 5, 6, and 7 shows the results for $(M^* + C^*)$, $(M_a^* + C^*)$, and $(M + C^*)$ methods, respectively, on all the 5 individual permutations of the modified bAbI dialog task test set.

User Accuracy		**Model ratio**
Per-turn	per-dialog	
Baseline method (M)		
81.73	3.7	100.0
Reward: 1, 2, -4, $(M + C^*)$		
92.44	33.1	50.98
92.57	32.7	53.22
93.96	42.0	44.63
94.7	43.0	45.82
90.59	16.6	65.24
Reward: 1, 3, -3, $(M + C^*)$		
90.41	21.1	54.67
92.25	36.1	54.79
92.12	32.5	58.07
89.75	18.2	65.69
92.05	24.6	60.92

Table 7: Test results for $M + C^*$ method on different permutations of modified-bAbI dialog task's test set.