End-to-end memory networks with word abstraction and contextual numbering for goal-oriented tasks

Asuka Sakai, Hongjie Shi, Takashi Ushio, Mitsuru Endo

AI Solution Center, Panasonic Corporation, Tokyo, Japan

{sakai.asuka1, shi.hongjie, ushio.takashi, endo.mitsuru}@jp.panasonic.com

Abstract

An end-to-end dialog system is useful because it does not need to make assumptions about the domain or dialog state structure. Additionally, it can achieve high versatility with only datasets. The sixth Dialog System Technology Challenge (DSTC6) introduced an end-to-end goal-oriented dialog task, in which several crucial skills are tested for a dialog system to achieve the goal of restaurant reservations. In this paper, we propose a method that combines end-to-end memory networks with the abstraction of named entities and contextual numbering to make the model effective in dealing with out-of-vocabulary words and learning context in a dialog. An evaluation with DSTC6 shows that our methods can effectively improve performance when the system must generate an API call to query the knowledge base or propose options. We consider that a system incorporating our methods can be scaled up to a new domain with only a knowledge base.

Index Terms: Memory networks, goal-oriented, end-to-end dialog systems

1. Introduction

Goal-oriented dialog systems have attracted much attention because of their usefulness. Young [1] outlined the typical structure of a goal-oriented dialog system. It consists of several independently developed modules: natural language understanding, dialog state tracker, and natural language generation. However, goal-oriented dialog systems have limitations because each module needs to be designed individually, and it is difficult to scale to new domains [2]. Thus, an end-to-end dialog system in which all components are trained from the dialogs themselves is required.

One of the tasks in the sixth Dialog System Technology Challenge (DSTC6) is an end-to-end goal-oriented dialog task, which is based on that of Bordes et al. [3]. It involves a restaurant reservation system. With this task, an end-to-end dialog system can be easily evaluated in terms of crucial capabilities such as performing dialog management, querying knowledge bases (KBs), interpreting the output of such queries to continue the conversation, and dealing with new entities.

In this paper, we describe an end-to-end memory network model with the replacement of named entities and numbering taking context into account. Section 2 introduces the dataset and the end-to-end goal-oriented dialog task of DSTC6. Section 3 describes an overview of our methods. In Section 4, the evaluation results are presented with their analysis. In Section 5, the conclusion drawn from this work are described.

2. Task and dataset description

In the end-to-end goal-oriented task, there are five subtasks to test the above-mentioned capabilities. All subtasks are described in Table 1. In every subtask, answers are chosen from candidates, and candidates are ranked. Additionally, each subtask requires solving all previous tasks.

The dialog simulation is based on KBs, the facts of which contain the restaurant information. Each restaurant is defined by a type of cuisine, a location, a price range, a rating, dietary restrictions, and atmosphere, and has an address and a phone number. The KB can be queried using API calls, which return the list of facts related to the corresponding restaurants. Each API call has a certain number of slots.

There are two types of KBs: KB1 and KB2. The difference between the two KBs is the slot value of the location and the type of cuisine, as described in Table 2. For training datasets, only KB1 can be used; thus, values in the two types of slots of KB2 are out of vocabulary (OOV).

Table 1: Subtask description

Task	Title	Request
1	Issuing API calls	Asking questions for filling the missing required fields and eventually generating the corresponding API call
2	Updating API calls	Asking users if they are done with their updates and issuing the updated API calls
3	Displaying options	Proposing options to users by listing the restaurant names sorted by rating until users accept
4	Providing extra information	Learning to use KB facts of restaurant information, such as the phone number of the restaurant, its address, or both, correctly to answer
5	Conducting full dialogs	Combining tasks 1-4 to generate full dialogs

Table 2: Slot value

Topic	Slot
Atmosphere	business, casual, romantic
Cuisine	(KB1) Italian, British, Indian, French, Spanish (KB2) Korean, Thai, Japanese, Cantonese, Vietnamese
Location	(KB1) Rome, London, Bombay, Paris, Madrid (KB2) Seoul, Bangkok, Tokyo, Beijing, Hanoi
Number	two, four, six, eight
Price	cheap, moderate, expensive
Restriction	vegan, vegetarian, gluten-free

There are four sets of test dialogs. Test1 has the same KB as for the training dialogs and the same set of slots in the queries; Test2 has the other KB and the same set of slots in the queries; Test3 has the same KB as for the training dialogs and one additional slot, which describes dietary restrictions for the queries; Test4 has the other KB and an additional required slot, which is the same as that of Test3. Test datasets include 1,000 dialogs, and performance is evaluated based on the accuracy rate of those dialogs.

3. Methods

For DSTC6, we propose a method based on the end-to-end memory networks originally proposed by Sukhbaatar [3]. It is a neural network with a recurrent attention model over multilayered memory, which can assume the next utterance by serially reading from and writing to the memory related to a query. This model is characterized by its high performance for the restaurant reservation task [3] and high simplicity because it can scale up to new domains. Thus. it is applied to our methods.

In this section, we describe an overview of the end-to-end memory network model and its evaluation. In addition, we propose our methods to improve the performance of this model for the restaurant reservation task.

3.1 End-to-end memory networks

The structure of end-to-end memory networks consists of several parts, including the part that stores and represents the dialog history, places attention on the memory, and chooses the response. The structure of the model is described in Fig. 1: (a) describes a single-layer version of the model, while (b) describes a three-layer version of the model. First, the embedded dialog history is represented in input memory and output memory. Next, the match between the embedded question sentence and the memories is computed by taking the inner product followed by a softmax, and it is defined as a probability vector over the inputs. The memory can be iteratively reread to search for additional pertinent information using the updated state, as shown in Fig. 1 (b). Finally, with the weight and output memory, certainty is computed for each candidate.

The evaluation of end-to-end memory networks was examined with training data consisting of 10,000 dialogs, which proved that there is room for improvement in performance. To reduce error cases, our methods are introduced as described below.

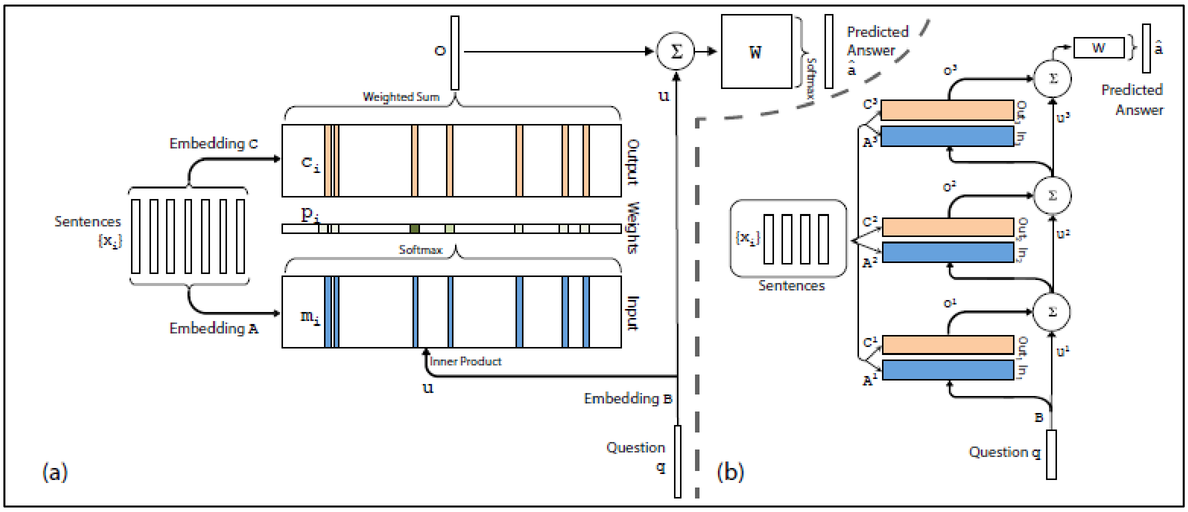

Figure 1: Overview of end-to-end memory networks

Table 3: Example of slot replacement

Data (Original)	**Data (with replacement)**
[Dialog History] U: Good morning. S: Hello, what can I help you with today? U: Can you book a table with **vegan** menu options in a **moderate** price range for **eight** people? S: I'm on it. U: <silence> S: Are you looking for a specific atmosphere? U: I'm looking for a **casual** atmosphere. S: Any preference on a type of cuisine? U: I got **Italian** last time, so I may be due for a change. S: Whenever you're ready. U: I think I had **French** last time, so let's do **British** food. S: Where should it be? U: I'm very fond of **Madrid**, but it's also good to change sometimes. S: Whenever you're ready. U: Let's try **London**, **Bombay** is out of the way for me. S: OK let me look into some options for you. U: <silence>	[Dialog History] U: Good morning. S: Hello, what can I help you with today? U: Can you book a table with **restrictions_1** menu options in a **price_1** price range for **number_1** people? S: I'm on it. U: <silence> S: Are you looking for a specific atmosphere? U: I'm looking for a **atmosphere_1** atmosphere. S: Any preference on a type of cuisine? U: I got **cuisine_1** last time, so I may be due for a change. S: Whenever you're ready. U: I think I had **cuisine_2** last time, so let's do **cuisine_3** food. S: Where should it be? U: I'm very fond of **location_1**, but it's also good to change sometimes. S: Whenever you're ready. U: Let's try **location_2**, **location_3** is out of the way for me. S: OK let me look into some options for you. U: <silence>
[Answer] S: api_call british london eight moderate casual vegan	[Answer] S: api_call cuisine_3 location_2 number_1 price_1 atmosphere_1 restrictions_1
[Candidates] ・api_call british london four moderate casual vegan ・api_call indian madrid eight moderate casual vegan ・api_call italian rome eight moderate casual vegan ・api_call british london eight moderate casual vegan ・Let's try London, Bombay is out of the way for me. ・Can you book a table with vegan menu options in a moderate price range for eight people? ・api_call french london eight moderate business vegan ・api_call british bombay eight moderate casual vegan ・api_call british london eight moderate business vegetarian ・api_call french london eight moderate casual vegan ・I'm looking for a casual atmosphere.	[Candidates] ・<not answer> ・<not answer> ・<not answer> ・api_call cuisine_3 location_2 number_1 price_1 atmosphere_1 restrictions_1 ・Let's try location_2, location_3 is out of the way for me. ・Can you book a table with restrictions_1 menu options in a price_1 price range for number_1 people? ・<not answer> ・api_call cuisine_3 location_3 number_1 price_1 atmosphere_1 restrictions_1 ・<not answer> ・api_call cuisine_2 location_2 number_1 price_1 atmosphere_1 restrictions_1 ・I'm looking for a atmosphere_1 atmosphere.

3.2 Word abstraction and contextual numbering for slot replacement

In Task 1, the system occasionally failed in generating an API call when the users mentioned several slot values in one utterance. It is considered that these errors occurred because the model cannot correctly learn the word order for multiple items in the user request. To solve these problems, the replacement of slot values was applied to the model. There are two parts in slot replacement. First, we generate a dictionary from KB, then, named entities are replaced with an abstract representation using it. For example, "Madrid" is replaced with "location" using the dictionary of location slot. Second, the number corresponding to order of appearance for each slot is added to slot name. For example, "Madrid" is replaced with "location_1", "London" with "location_2", and "Bombay" with "location_3" as indicated in Table 3.

3.3 Word abstraction and contextual numbering for replacement of restaurant name

In Task 3, the KB is queried using the corresponding API call, and the resulting facts are added to the dialog history. The system must propose restaurants sorted by their rating in decreasing order until users accept. In error cases, the system fails to repeat the name of the restaurant mentioned before. In order to solve this problem, we replace the restaurant names with their rating rank so that the dialog system can handle the restaurant names only using indexes. For example, the restaurant name was replaced with "cand_" + "rating rank" as "cand_1," "cand_2," and "cand_3" in descending order of rating value. Restaurant list in Table 4 indicates options that the system proposes along the user's request. Then, we replaced the restaurant name using rating rank as indicated in Table 5.

Table 4: Comparison table of replacement

Restaurant name	Rating	With replacement
gentle_calm_moment_right	57	cand_2
the_broad_calm_fish_issue	58	cand_1
the_clean_zealous_side_city	12	cand_4
the_adorable_important_horse_head	44	cand_3

Table 5: Example of restaurant replacement

Data (original)	Data (with replacement)
[Dialog History] U: Hello. S: Hello, what can I help you with today? U: May I have a table for six people in Seoul with Korean food with a casual atmosphere in a cheap price range?. S: I'm on it. U: <silence> S: OK let me look into some options for you. U: <silence> S: What do you think of this option: **the_broad_calm_fish_issue** ? U: No, I don't like it. S: Sure let me find another option for you. U: <silence> S: What do you think of this option: **gentle_calm_moment_right**? U: What was the first one you suggest? S: The option was **the_broad_calm_fish_issue**. U: I still don't like that. S: Sure let me find another option for you. U: <silence>	[Dialog History] U: Hello. S: Hello, what can I help you with today? U: May I have a table for six people in Seoul with Korean food with a casual atmosphere in a cheap price range?. S: I'm on it. U: <silence> S: OK let me look into some options for you. U: <silence> S: What do you think of this option: **cand_1**? U: No, I don't like it. S: Sure let me find another option for you. U: <silence> S: What do you think of this option: **cand_2**? U: What was the first one you suggest? S: The option was **cand_1**. U: I still don't like that. S: Sure let me find another option for you. U: <silence>
[Answer] S: What do you think of this option:**the_adorable_important_horse_head** ?	[Answer] S: What do you think of this option: **cand_3** ?
[Candidates] ・What do you think of this option: the_green_bad_power_head? ・What do you think of this option: **the_broad_calm_fish_issue**? ・What do you think of this option: **the_clean_zealous_side_city**? ・What do you think of this option: the_able_steep_state_world? ・What do you think of this option: the_adorable_straight_case_business? ・What do you think of this option: the_able_crooked_head_area? ・What do you think of this option: **gentle_calm_moment_right**? ・What do you think of this option: the_important_faithful_face_idea? ・What do you think of this option: the_agreeable_different_work_name? ・What do you think of this option: the_drab_great_head_chicken ・What do you think of this option: **the_adorable_important_horse_head**?	[Candidates] ・What do you think of this option: the_green_bad_power_head? ・What do you think of this option: **cand_1**? ・What do you think of this option: **cand_4**? ・What do you think of this option: the_able_steep_state_world? ・What do you think of this option: the_adorable_straight_case_business? ・What do you think of this option: the_able_crooked_head_area? ・What do you think of this option: **cand_2**? ・What do you think of this option: the_important_faithful_face_idea? ・What do you think of this option: the_agreeable_different_work_name? ・What do you think of this option: the_drab_great_head_chicken ・What do you think of this option: **cand_3**?

Similarly, in the error cases of Task 4, the system occasionally proposed the address or phone number of incorrect restaurants. Thus, the restaurant name was replaced as in Task 3.

3.4 Removal of candidates

To improve performance further, candidates were narrowed when an API call is included in a candidate sentence. Candidates that included slot values not appearing in the dialog history were replaced with "<not answer>," as indicated in the candidate list in Table 3.

Table 6: Evaluation of our model

	Test1	Test2	Test3	Test4
Task 1	0.998	0.997	0.946	0.957
Task 2	1.0	1.0	0.932	0.928
Task 3	0.991	0.993	0.993	0.997
Task 4	0.995	0.997	0.996	0.996
Task 5	0.986	0.984	0.928	0.932

Table 7: Hyperparameters of memory networks

Task	Learning Rate	Epoch	Embedding Dim.	Hops
1	0.01	200	32	2
2	0.01	200	32	2
3	0.01	200	32	3
4	0.01	200	32	3
5	0.01	200	32	3

4. Results

The results of our proposed methods, along with the precision scores, are listed in Table 6. Each score indicates the probability that the estimated candidate sentence matches the correct answer. The scores of all cases were greater than 0.9. Hyperparameters of the model submitted are summarized in Table 7. Here, to investigate how much our proposed methods contribute to the results, the accuracy was compared between cases with slot/restaurant replacement and without it. Accuracy was calculated using test data and answers we predicted using our in-house rule-based dialog tracking model with hand-crafted dialog states. We assessed the answers predicted using this model to be accurate because the accuracy obtained using the model is equal to that of distributed results. All comparison results are shown in Fig. 2. For Task 2, we did not verify the contribution of our proposed methods because the model can achieve a high prediction performance even without our methods.

4.1 Effect of slot replacement

A comparison of results between memory networks with the replacement of slot value and without it is shown in Fig. 2 (a). In Task 1, performance can be improved with our methods, especially when OOV words were included in test datasets. Similarly, Fig. 2 (d) shows that slot replacement and contextual numbering were effective in Task 5; however, the replacement of restaurant name also contributed to this result. The replacement of slot value can round variations of slot values. Moreover, each slot value in one dialog can be distinguished by numbering taking context into account.

4.2 Effect of restaurant replacement

A comparison of results between memory networks with the replacement of restaurant name and without it is also shown in Fig. 2. In Task 3, performance can improve to nearly perfect, as shown in Fig. 2 (b). Similarly, Fig. 2 (c) and (d) show that our methods were effective in Task 4 and Task 5. The abstraction of slot value implies that large variations in restaurant name can be rounded to achieve effective use of limited training data. Additionally, the model can easily learn which restaurant to propose by adding the rating rank.

4.3 Additional Slot

For Test 3 and Test 4, we examined the pre-processing of training data. In these cases, one slot was added to test data; thus, corresponding dialog data were added. For example, question such as “do you have any dietary restrictions?” and answers such as “vegan” were added before the API call of the system. We considered that our methods are effective for learning new slots; however, it failed in improving performance. Thus, answers were estimated with the model as in Test 1 and Test 2. In fact, our results for Test 3 and Test 4 revealed that almost all errors occurred when the system asks the user about dietary restrictions.

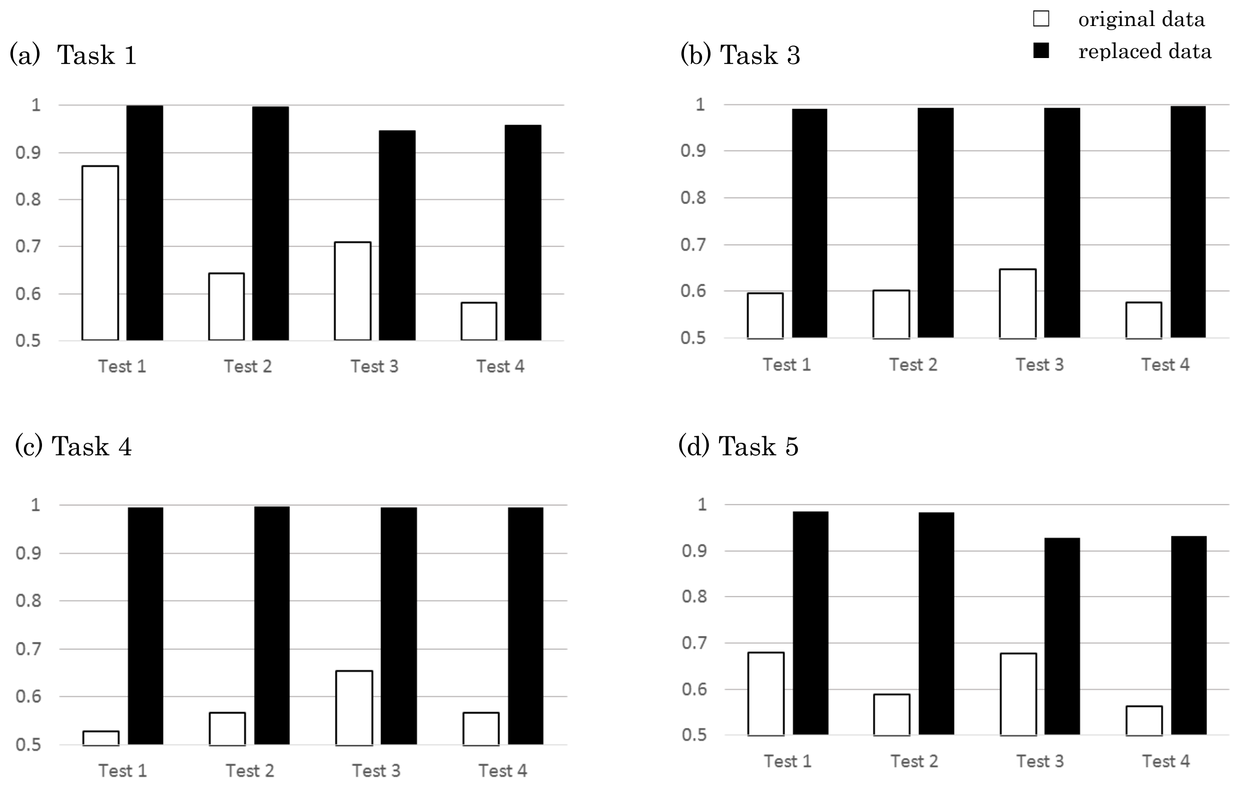

Figure 2: Prediction accuracy of model with/without replacement

5. Conclusion

In this paper, we proposed end-to-end memory networks with the abstraction of proper nouns such as restaurant name and slot value as well as numbering taking dialog context into account. Evaluation based on DSTC6 showed that our methods are effective to improve performance in the case where the system must generate an API call, propose a restaurant, and offer adequate information on it. Notably, we can deal with OOV words almost perfectly by only adding those new words to KB. Thus, our methods can make it easy to extend to a new domain. It can be said that our method is useful for building goal-oriented dialog systems such as a restaurant reservation system and food ordering system.

6. References

[1] S. J. Young, “Using pomdps for dialog management,” in SLT 2006, pp. 8-13.

[2] T. Zhao, M. Eskenazi, “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning,” in Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2016.

[3] A. Bordes, Y.-L. Boureau, J. Weston, “Learning end-to-end goal-oriented dialog,” in ICLR 2017.

[4] S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, “End-to-end memory networks,” in NIPS, 2014.