A Corpus-free State2Seq User Simulator for Task-oriented Dialogue

Yutai Hou[†], Meng Fang[‡], Wanxiang Che[⋆†], Ting Liu[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{ythou, car, tliu}@ir.hit.edu.cn
[‡] Tencent Robotics X, mfang@tencent.com

Abstract. Recent reinforcement learning algorithms for task-oriented dialogue system absorbs a lot of interest. However, an unavoidable obstacle for training such algorithms is that annotated dialogue corpora are often unavailable. One of the popular approaches addressing this is to train a dialogue agent with a user simulator. Traditional user simulators are built upon a set of dialogue rules and therefore lack response diversity. This severely limits the simulated cases for agent training. Later data-driven user models work better in diversity but suffer from data scarcity problem. To remedy this, we design a new corpus-free framework that taking advantage of their benefits. The framework builds a user simulator by first generating diverse dialogue data from templates and then build a new State2Seq user simulator on the data. To enhance the performance, we propose the State2Seq user simulator model to efficiently leverage dialogue state and history. Experiment results on an open dataset show that our user simulator helps agents achieve an improvement of 6.36% on success rate. State2Seq model outperforms the seq2seq baseline for 1.9 F-score.

1 Introduction

Task-oriented dialogue systems assist users to achieve specific goals such as finding restaurants or booking flights [25]. To learn such a system is very challenging. Recently, reinforcement learning (RL) methods have been introduced due to their advantages in sequential decision. [25, 18, 24, 7]. An RL based dialogue agent can learn from dialogue data or reward signals by interacting with real users. Unfortunately, interacting with real users is costly and there is often no enough data or even no data for new domains. To overcome these obstacles, building user simulators is studied for training RL dialogue algorithms [22, 14].

User simulators can be divided into two categories: traditional and data-driven user simulator. Traditional user simulators are agenda-based or rule-based [13, 20]. A rule-based user simulator can be built without data, but needs domain-specific knowledge and hard to generalize to new contexts. Besides, the rule-based model lacks response diversity, which largely limits the effectiveness of RL

⋆ Email correspondence.

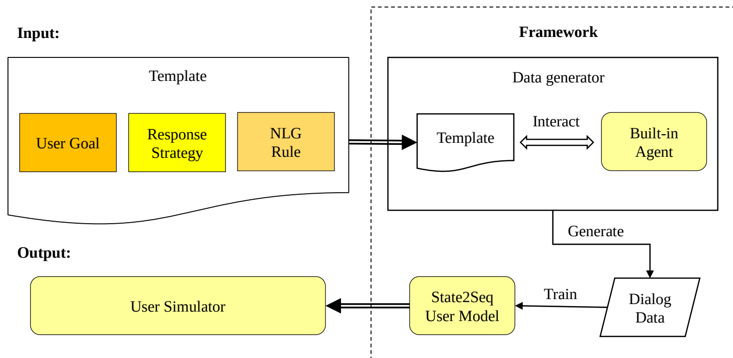


Fig. 1. The proposed corpus-free framework for building user simulators.

training. Latter data-driven user simulators ease the problem of diversity and depend less on expert knowledge. They imitate user behaviors from datasets with statistical models, such as Bayesian models [17, 8], hidden Markov models [4] and seq2seq models [1]. Statistical user simulators are inherently diverse and often require a large amount of expert-labeled data for training. However, they can not cope with limited data situations.

In this paper, to overcome the data scarcity problem and build a user simulator with sufficient diversity, we propose a new corpus-free framework for building user simulators. It combines the ideas of rule-based and data-driven user simulators. As shown in Figure 1, the framework generates dialogue data from templates and train data-driven user model on it. The template consists of user goals, response strategy and natural language generation rules. An example of the template is shown in Table 1. In addition to templates, data generation uses an RL-based built-in agent to improve data diversity and explore more dialogue cases. The statistical nature of data-driven user simulator provides more diversity than rule-based ones. Diverse responses allow covering more situation for policy training.

To enhance the user simulator's quality, we propose a novel attention based State2Seq user simulator to leverage the dialogue state and history better. The model first learns representations for dialogue context items. Dialogue context contains structured data of dialogue states, user goals and agent response. Then for each dialogue turn, the model predicts user actions sequentially with the attention on context. Attention helps to pick action more accurately. For example, suppose agent response is (request:movie, inform:date=today), and user goal is (request=[ticket], constraint=[date:tomorrow, movie:Deadpool]). The user model can easily output actions (inform:movie, deny:date) by attending to the agent request and the states of constraint inconsistency respectively.

Experiments are conducted on the movie booking dataset [13] and an in-house restaurant domain dataset. We evaluate both the user simulator model itself and the policy trained by it. On movie booking dataset, our policy achieves

Table 1. Template example for movie booking domain. G is user goal, V is response strategy and N is NLG rules. `req` is request.

Template Name	Template Content
G	$g_0 = \begin{bmatrix} C = [\texttt{movie} = \text{Godfather}, \texttt{time} = 5\text{ pm}], \\ R = [\texttt{ticket}, \texttt{theater}] \end{bmatrix}, g_1, g_2, \ldots$
V	$r_0 = \begin{bmatrix} \textbf{if } A_{t-1} = \{\texttt{req:time}\} \text{ and } \texttt{time} \text{ in } g.C \\ \textbf{then } A_t = \texttt{inform:time} = g.C.\texttt{time} \end{bmatrix}, r_1, r_2, \ldots$
N	$l_0 = \begin{bmatrix} \textbf{if } A = \{\texttt{inform:time}=\text{g.C.}\texttt{time}\} \\ \textbf{then } L = \text{"I want to see it at 5 pm"} \end{bmatrix}, l_1, l_2, \ldots$

an improvement of 6.36 points on the success rate over the strongest baseline. And proposed State2Seq model outperforms the seq2seq baseline for 1.9 F-score on response accuracy.

This paper has 3 main contributions: 1. To solve data scarcity, we design a new corpus-free framework for building a user simulator with response diversity. 2. We introduce the attention mechanism to task-oriented user simulator and propose a State2Seq model to get better user behavior modeling. 3. Experiments show that response diversity and attention on dialogue context improve user model and agent policy.

Our code is available at: https://anonymous.com

2 Proposed Framework

We focus on developing a user simulator, which is diverse to cover enough dialogue situations. To solve the data scarcity problem, our framework builds a user simulator with only templates and no dialogue data. There are two main components in the framework: a template based data generator and a neural user simulator. The framework (1) first generate data from the templates using a built-in agent, (2) then train a data-driven user simulator on it.

2.1 Template Definition

Template T is the input to our framework and is used to generate data. We define a template as: $T = (G, V, N)$, which includes user goals G, response strategies V and natural language generation rules N. An example of movie booking domain template is illustrated in Table1.

G is a set of predefined user goals and defined as: $G = \{g_i\}_i^\alpha$, where α is the number of goals. Each user goal g is defined as $g = (C, R)$ which includes a set of user constraints C and a set of user requests R [20].

V is a set of rules for response strategies, which is relatively easy to obtain [13]. It is defined as: $V = \{r_i\}_i^\beta = V_a \cup V_u$, where V_a and V_u are response rules for user and agent respectively. For each $r \in V$, we define it as a function that

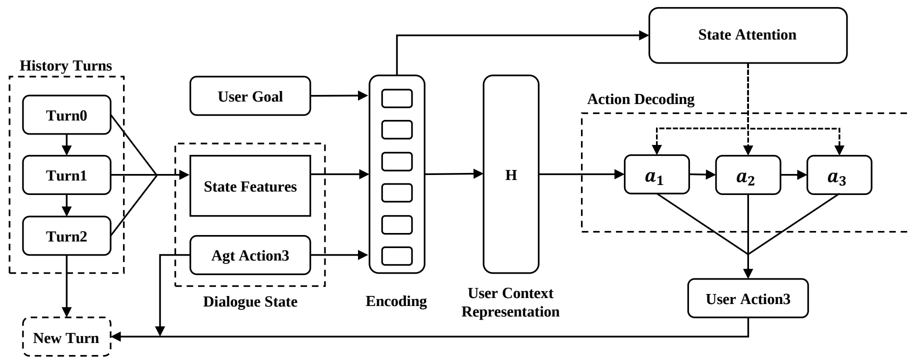


Fig. 2. State2Seq user simulator.

maps dialogue context to response:

$$\begin{cases} r: f(A_1, A_2, ..., A_{t-1}) \rightarrow A_t, \quad\quad r \in V_a \\ r: f(A_1, A_2, ..., A_{t-1}, G) \rightarrow A_t, r \in V_u \end{cases}$$

where A_i is the response for the i_{th} turn. Specifically, as User and agent may take multiple actions in one turn, we define response A as a set of single actions: $A = \{a_i\}_0^k$.

$N = \{l_i\}_i^\gamma$ is a set of rules for natural language generation(NLG): Each rule l maps actions and user goal to natural language L and is defined as $l : f(a_1, a_2, a_3, ..., a_n, g) \rightarrow L$.

2.2 Data Generation

The data generator generates conversation log with templates T as input, as shown in Figure 1. There are two steps for the generation: i. Generate basic data with templates only. ii. Generate diverse data with a built-in RL agent and templates.

Generate basic data To start from no corpus, we first collect some rule-based conversation as basic data, which is a common warm-up option [13]. When generating one dialogue, we first pick a user goal g from G as background and construct a random starting utterance. Then, for each turn, we search the suitable rule $r \in V$ to generate the response actions for user and agent. NLG rules N is then used to render actions to utterance.

Generate diverse data On the basis of basic data, we generate diverse data. A built-in RL based agent $\mathcal{M}^*$ is used here. $\mathcal{M}^*$ is warmed up with basic data initially and further trained by interacting with V_u. When generating each dialogue, all process are same to basic data generation, except for agent response actions are given by $\mathcal{M}^*$'s policy.

We enhance the data diversity by the following operations: i. Leverage the RL exploration mechanism during data generation. ii. Generate data with the

Table 2. Feature definition

Feature	Description
Constraint Status	Status about whether user constraint slots have been informed by user
Request Status	Status about whether user request slots have been satisfied by agent
Slot Consistency	Status of whether the slot values provided by agent are consistent to user constraints
Dialog Status	Dialogue status of success, failed and no outcome yet

built-in agents of different training stages. So both weak and strong policy are used, which allows collecting both clumsy and fluent dialogue.

2.3 User Model Training

After collecting the dialogue data, we train the user simulator on it in supervised style. Given a dialogue context, the user model is trained to predict a set of user actions as response. Only user policy is learned from data here.

3 State2Seq User Simulator

We aim to propose a user simulator that makes better use of dialogue context.

User simulator mimics human responses to the dialogue system output. A user model predicts user response with dialogue context. Given user goal g and dialogue history $< A_{t-1}, A_{t-2}, ..., A_1 >$, it predict t_{th} turn response as :

$$A_t = \arg\max_A p(A \mid g, A_{t-1}, A_{t-2}, ..., A_1)$$

where $A = \{a_i\}_0^k$. Following [1], we formulate the action selection as a sequence generation problem:

$$p(A \mid g, A_{t-1}, A_{t-2}, ..., A_1) = \prod_i^n p(a_i \mid a_1, ..., a_{i-1}, g, A_{t-1}, A_{t-2}, ..., A_1)$$

However, dialogue history can be very long. So it is very hard for the user model to leverage history information directly. To remedy this, we extract the key information as dialogue state S from the dialogue history. We then define the dialogue context as a combination of dialogue state and user goals.

For better usage of dialogue context, we learn vector representations for each context items. And attention mechanism is proposed to leverage context more clearly.

Figure 2 shows the structure of the State2Seq user simulator. The State2Seq model maintains a dialogue state S for tracking the dialogue history. It uses an encoding module to provide vector representations for dialogue context items. For each turn, the model integrates those representations into a context representation **H**. Then the model decodes **H** into a sequence of actions as output. Attention helps the model use dialogue context during decoding.

3.1 Dialogue Context Representation

The main idea of dialogue context representation is to refine history information. Forgetting useless information is proven to be important for data-driven model.

The dialogue context consists of dialogue state and user goal. Dialogue state S includes last turn agent response and state features. Following [1], we extract the state features explained in Table 2. Constraint Status, Request Status, and Dialog Status are used to help the user simulator track the progress of the current conversation, and Slot Consistency allows the user simulator to correct the agent on wrong information. Each feature is recorded as a status vector $\{status\}_0^m$, where m is the number of slot types and $status$ could be 1, 0, -1 for active, irrelevant and negative.

Attention mechanism relies on a good representation of items. Representation is also important for sequence decoding, as its need a good initial state **H**. So we propose an encoding module to learn vector representations for context items. To help to learn of representation, we share representations for the common slots in context items. Some negative status is rare in the corpus, which makes it hard to learn good representations. To remedy this, we only learn vector representations $\mathbf{e}$ for positive slot status, and then use the corresponding inverse vector $-\mathbf{e}$ to represent negative status.

H is obtained by dimension reduction of context representation, which can further forget irrelevant information. Formally, given a agent response A_{t-1}, the user goal g and current dialogue history $< A_{t-1}, A_{t-2}, ..., A_1 >$, State2Seq updates dialogue state S and represent context with vectors. **H** is then obtained as:

$$\mathbf{H} = \mathbf{W}_c \cdot ([\mathbf{E}(A_{t-1}); \mathbf{E}(g); \mathbf{E}(S)]) + \mathbf{b}_c$$

where $[;]$ denotes vectors concatenation and $\mathbf{E}(\mathbf{x})$ means fetching vector representations for items in x.

3.2 Action Generation with Attention

We formulate actions selection as sequence generation process with attention. Sequence generation provides diversity in selecting actions. Attention mechanism helps to use dialogue context information directly. Specifically, we generate responses by attending on items in user goal g, dialogue state S and the last agent response A_{t-1}.

During actions decoding, for each time step t, LSTM provides hidden representation $\mathbf{h_k} = \text{LSTM}(\mathbf{h_{k-1}}, \mathbf{c_{k-1}}, \mathbf{x_k})$, where $\mathbf{h_k}$ denotes the LSTM hidden state at time step k, the $\mathbf{c_{k-1}}$ is cell state and $\mathbf{x}$ is input.

The attention weight of i_{th} item in dialogue context is calculated as: $\mathbf{att_i} = \frac{\exp(\mathbf{e_i} \cdot \mathbf{h_k})}{\sum_j \exp(\mathbf{e_j} \cdot \mathbf{h_k})}$ where $\mathbf{e_i}$ is semantic representation vector of i_{th} item. Attention is used to calculate the decoding output $\mathbf{h'_t} = \tanh(\mathbf{w_a} \cdot (\mathbf{att} \cdot \mathbf{Emb}) + \mathbf{b_a} \cdot \mathbf{h_k})$, where $\mathbf{Emb}$ is the representations of all items in context.

Then we model the distribution the time step k's action a_k:

$$P(a \mid g, S, A_{t-1}, a_{k-1}, ..., a_1) = \mathrm{Softmax}(\mathbf{W_p} \mathbf{h'_t} + \mathbf{b_p})$$

And the user action a_k is predicted as:

$$a_k = \arg\max_a p(a \mid g, S, A_{t-1}, a_{k-1}, ..., a_1)$$

4 Experiment

4.1 Dataset

We used two datasets in our experiments: movie ticket booking data and restaurant reservation data. The movie ticket booking data is an open task-completion dialogue dataset proposed by [13]. For each dialogue, the system gathers information about the customer's desires and books the movie tickets. The success or failure of dialogue is assessed based on (1) whether a movie is booked, and (2) whether the movie satisfies the user's constraints. The data includes 11 dialogue acts, 29 slots, 277 user goal templates. Rule templates for response strategy and NLG are also included in [13]'s work.

To test the method's generation ability, we also build a dataset for restaurant reservation domain. In each dialogue, the user reserves a table under his/her requirements. The data includes 11 dialogue acts, 24 slots and 184 user goal templates. We design rule templates for response strategy and NLG based on the ones in [13]'s work.

4.2 Evaluation

We evaluate a user simulator by: i. evaluation of the user simulator itself. ii. evaluation of the policy trained with it.

Evaluation of Agent Policy The main value of simulator is to train agent policy. We use both human and automatic evaluation for policy here. A DQN model is used as the agent to learn the policy.

We adopt cross-model evaluation proposed by [19]. $\mathcal{N}$ user models are first used to train $\mathcal{N}$ policies. Each policy is then tested against $\mathcal{N}$ different user simulators. Finally, we calculate the average of $\mathcal{N} \times \mathcal{N}$ scores. A policy trained by a good simulator can still perform well on poor simulators [19, 11]. A higher average score indicates a better simulator ability for training agent. For metric, we use success rate, average reward and average turn number, which have been widely accepted as a standard metric of multi-turn agent [6, 2, 13].

Evaluation of User Simulator We evaluate the user simulator itself from two aspects.

Firstly, following [1], we evaluate the accuracy of predicting actions. F-score is used as the metric.

Secondly, a user simulator's generalization ability is also important. It has more tolerance for exploration of training agent. We compare different user simulators' such ability by making conversation against a same rule agent. User model with better ability should achieve more success rate.

4.3 Model Details

For the State2Seq model, we set embedding size as 300. We use a 2 layer LSTMs for decoding with hidden state size of 256. During the training, we set the batch size as 32, a dropout as 0.8 and teacher forcing rate of 0.5. The learning rate is set as 0.001 and we set a learning rate decay of 0.9.

For the RL agent model used in data generation and evaluation, we use the DQN model. We set experience pool size as 1000, hidden layer size as 80. Experience replay redesigned for dialogue setting is applied. We use ϵ-greedy exploration of 0.01. The learning batch size is 16. We use 100 warm-up epochs and 500 training epochs. Model simulates 100 dialogues for each epoch.

We also simply extend our model by replacing the sequence decoder with a multi-label classifier. We name it as State2MLC. State2MLC takes dialogue state, user goal, agent response as input and predict multiple actions. State2MLC is trained with Multi-Label Soft Margin Loss.

4.4 Baselines

We compared with the following baselines:

- **Seq2Seq** is user simulator proposed by [1]. It extracts history turns' features as input sequence and decodes action sequence.
- **Seq2Seq-att** is based on Seq2Seq model and adds attention mechanism over input sequence.
- **Agenda** based user simulator is proposed by [20]. It is corpus free and generates user response by maintaining a user agenda with rule. We use the agenda user simulator provided by [13].

4.5 Performance of Agent Policy

We compare the policy trained by different user simulators with cross-model evaluation.

Results on movie booking data Table 3 shows the evaluation results on movie booking domain. The results show that the policy obtained by our model outperforms baselines.

On average success rate, policy trained by our State2Seq model outperforms the Agenda model for 6.36 points. As the Agenda [13] model is rule-based, the main difference between State2Seq and it is that State2Seq has more diverse responses. This demonstrates that user simulator diversity improves policy ability for finishing task and generalization. Policy trained by our model outperforms the Agenda model on success rate and average reward.

Table 3. Evaluation of agent policy trained by different user model. Results above dash-line are from our model, which achieve best performance in most task.

Model	Movie Booking			Restaurant Reservation		
	Avg. Succ.	Avg. Rwd.	Avg. Turns	Avg. Succ.	Avg. Rwd.	Avg. Turns
State2MLC	0.487	8.55	**21.82**	0.305	-17.22	29.69
State2Seq	**0.551**	**14.17**	25.85	**0.524**	**11.77**	24.21
Seq2Seq	0.412	-3.49	27.77	0.501	6.23	29.83
Seq2Seq-Att	0.430	-2.59	30.39	0.514	9.67	26.20
Agenda	0.438	-2.88	32.88	0.508	10.88	**22.17**

Table 4. Human evaluation of trained agent

Model	Avg. Succ.	Avg. Rwd.	Avg. Turns
State2Seq (Ours)	**0.778**	**53.88**	**11.88**
Agenda	0.571	22.29	14.57

The results show that the policy trained by other statistical methods underperform the Agenda based model. Because those user simulators lack for response accuracy, which will mislead and confuse the policy training. Comparing Seq2Seq-att to Seq2Seq, the results show the effectiveness of attention mechanism. And State2Seq's improvements over the Seq2Seq-att show that the refined context representation of the State2Seq model does help the response generation.

Results on restaurant data Table 3 shows the cross evaluation results for restaurant reservation domain. The results show our method could work consistently well in different domains. Most models score higher on data in the restaurant domain, because the field is relatively simpler and has fewer slots. The State2MLC model does not perform well. This is due to the fact that State2MLC model has a much simpler structure, so it is likely to overfit to generated data in a simple domain and limits the generalization ability of the policy.

Human evaluation We perform a human evaluation on the movie domain, and each agent is tested by chatting with 2 domain experts for 50 dialogues. Table 4 shows that the agents trained in our model can be better adapted to the real situation.

Analysis on policy training process Table 5 shows the evaluation of training process. We perform testing at each training epoch, and report the averaged score. The policy is evaluated against the environment for training. Policy trained by our model outperforms the ones from statistical user simulator, which reflects that our user simulator improves the training performance of agent policy. It is because our user simulator has better generalization to respond to agent's unreasonable actions in the early training stage. On the other hand, diversity helps RL algorithm training. Agenda achieve good scores as rule environment is relatively easy for overfitting.

Table 5. Analysis of agent policy performance during training on movie domain. At beginning of each policy training epoch, we test the policy's performance. The results reflect the models' overfitting to training set.

Model	Succ.	Avg. Rwd.	Avg. Turns
State2MLC (Ours)	0.628	26.19	16.36
State2Seq (Ours)	0.800	48.39	17.23
Seq2Seq	0.462	2.98	28.92
Seq2Seq-Att	0.480	2.11	32.98
Agenda	**0.814**	**50.36**	**15.66**

Table 6. Evaluation of user simulator model.

Model	Action Accuracy		Generalization Ability Test			
	Movie	Restaurant	Movie		Restaurant	
	F1	F1	Avg. Succ.	Avg. Rwd.	Avg. Succ.	Avg. Rwd.
State2MLC (Ours)	0.704	**0.695**	**0.442**	**6.06**	0.436	5.34
State2Seq (Ours)	**0.711**	0.683	0.400	5.51	**0.484**	**11.09**
Seq2Seq	0.692	0.662	0.063	-31.87	0.126	-31.87
Seq2Seq-Att	0.705	0.677	0.000	-46.99	0.000	-46.99
Agenda	N/A	N/A	0.392	0.04	0.410	2.20

4.6 Performance of User Simulator

User model's performance is mainly reflected by the ability of predicting user responses.

Action accuracy. Table 6 shows the models' accuracy of predicting user actions. Our model achieves the best performance, outperforming the seq2seq model [1] for 1.9 points on F-score. The results also show a correlation between the evaluation of action prediction and the agent policy's performance, which demonstrates that user simulator quality affects agents performance.

The improvements mainly come from two aspects: Firstly, the attention mechanism provides specific context information. Secondly, refined context representation filters the useless information. The Seq2Seq-att model outperforms the Seq2Seq model by adding the attention mechanism. This reflects the effectiveness of the attention mechanism. By comparing State2Seq to Seq2Seq-att, improvement shows that due to forgetting mechanism, refined context representation is more effective than sequence encoder.

Analysis of generalization ability The generalization ability is also important for a user simulator in agent training. We compare different user simulators' generalization by making them chat with a same rule-based agent. Table 6 shows the results and State2Seq and MLC2Seq are optimal in all user simulators. Our best model outperforms the Agenda for 5.0 and 7.4 points success rate on the 2 domains. As the user simulators are trained to mimic user rules in the

Table 7. Case study of the difference between our user simulator and agenda user simulator. Here, user is requesting for ticket and theater, and user's constraints are { movie name: deadpool, city: Seattle , num: 2, date: tomorrow }

Agenda User Simulator	
...	
usr: Which theater is available?	act:req, req slots: {theater}
agt: Which theater would you like?	act:req, req slots: {theater}
usr: Which theater is available?	act:req, req slots: {theater}
agt: Which theater would you like?	act:req, req slots: {theater}
(loop...)	
Our User Simulator	
...	
usr: Which theater is available?	act:req, req slots:{theater}
agt: Which theater would you like?	act:req, req slots:{theater}
usr: I want to watch at Seattle.	act:inform, inform slots:{city: Seattle}
agt: Seattle is available.	act:inform, inform: {city: Seattle}
usr: Which theater is available?	act:req, req slots: {theater}
agt: The Pearl Theater is available.	act:inform, inform slots: {theater: The Pearl Theater}
...	

template, the improvement reflects that our framework can generate new diverse dialogues data to avoid user simulator overfitting to user response strategy in the template. Other user simulators perform worse on this test, we address this to the fact that these methods are less accurate in generating user actions and rule-based agent has a low allowance for response error.

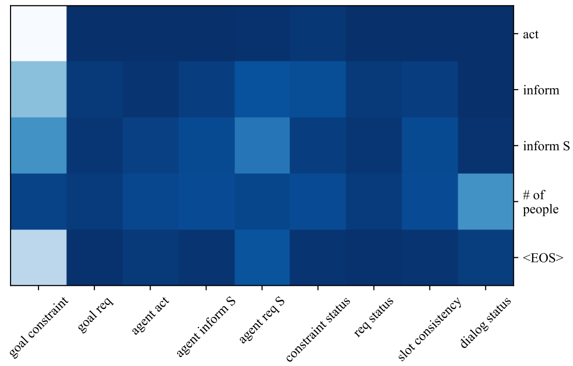
4.7 Case Study and Visualization

To find out the difference between rule-based model and the proposed model. We perform case study on dialogues between the user simulator and an agent. The comparison is shown in Table 7. When agents and users are unable to satisfy each other, the fixed rules of Agenda user simulator may be trapped in the loop shown in the example. But the response from our user simulator is diverse and uncertain. It can try other actions to jump out of the endless loop. The difference shows that our proposed framework for building user simulators has successfully improved the response diversity.

To demonstrate the effectiveness of the attention mechanism over context, we provide a visualization example shown in Figure 3. The figure shows that the attention mechanism successfully learns the correlation between the generated actions and the context. Specifically, when generating the dialogue action of `inform` and the inform slots, the proposed model pays a higher attention to context items of `sys_request_slots` and `user_goal`.

5 Related Work

There is little work solving the data insufficient problem of user simulator. [11] proposed a user simulator that generates user utterance directly, which could ease the effort of user semantic annotation. However, their work is corpus-based



Context:
agt:How many tickets do you need?
usr:I want 2 tickets please!

Actions:
act:req, inform S:{}, req S:{# of people}
act:inform, inform S:{# of people:2}, req S:{}

Fig. 3. Visualization of attention. The vertical axis is the generated user response and the horizontal axis is the dialogue context. The lighter color means higher attention. S denotes slots.

and still needs a large corpus to train on it. Other data scarcity problems for task-oriented dialogue system are also investigated. [26] leverage the idea of zero shot learning [16]. They solve the problem of dialogue generation for a new domain by mapping actions to latent space. [12] and [9] provided data augmentation methods for language understanding. [23] proposed to build new domain agent efficiently by machine self-play and crowdsourcing.

The first user simulator was proposed by [5]. There are two kinds of user simulators in terms of working levels. User simulators of semantic level interact to agent with dialogue acts and corresponding slot-value pairs. [21, 3, 1, 6]. User simulators of utterance level communicates to agent with utterance directly [10, 11, 15]. Our user simulator can work on both of the two levels.

6 Conclusion

In this paper, we study the problem of building user simulators for task-oriented dialogue from templates with no corpus. We solve the data scarcity and increase simulator response diversity by proposing a corpus-free framework. In our framework, we generate diverse data with only templates and trains a data-driven user simulator on it. To predict user response more accurately, we proposed a novel State2Seq user model. It predicts user response with attention on refined dialogue context representations. Experiment results show that with more response diversity, our user simulator improves the agent policy by 6.36% success rate. Attention and refined context representation help the State2Seq model outperform Seq2Seq baseline for 1.9 F-score.

Acknowledgments

We are grateful for helpful comments and suggestions from the anonymous reviewers. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61632011, 61772153 and 61772156.

References

1. Asri, L.E., He, J., Suleman, K.: A sequence-to-sequence model for user simulation in spoken dialogue systems. In: Proc. of Interspeech. pp. 1151–1155 (2016). https://doi.org/10.21437/Interspeech.2016-1175, https://doi.org/10.21437/Interspeech.2016-1175
2. Casanueva, I., Budzianowski, P., Su, P., Mrksic, N., Wen, T., Ultes, S., Rojas-Barahona, L.M., Young, S.J., Gasic, M.: A benchmarking environment for reinforcement learning based task oriented dialogue management. CoRR **abs/1711.11023** (2017), http://arxiv.org/abs/1711.11023
3. Chandramohan, S., Geist, M., Lefevre, F., Pietquin, O.: User simulation in dialogue systems using inverse reinforcement learning. In: Proc. of Interspeech (2011)
4. Cuayáhuitl, H., Renals, S., Lemon, O., Shimodaira, H.: Human-computer dialogue simulation using hidden markov models. In: Proc. of ASRU. pp. 290–295. IEEE (2005)
5. Eckert, W., Levin, E., Pieraccini, R.: User modeling for spoken dialogue system evaluation. In: Proc. of ASRU. vol. 97 (1997)
6. Gao, J., Wong, K., Peng, B., Liu, J., Li, X.: Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In: Proc. of ACL. pp. 2182–2192 (2018), https://aclanthology.info/papers/P18-1203/p18-1203
7. Gašić, M., Young, S.: Gaussian processes for pomdp-based dialogue manager optimization. IEEE/ACM Trans. on Audio, Speech & Language Processing **22**(1), 28–40 (2014)
8. Georgila, K., Henderson, J., Lemon, O.: Learning user simulations for information state update dialogue systems. In: Proc. of Eurospeech (2005)
9. Hou, Y., Liu, Y., Che, W., Liu, T.: Sequence-to-sequence data augmentation for dialogue language understanding. Proc. of COLING (2018)
10. Jung, S., Lee, C., Kim, K., Jeong, M., Lee, G.G.: Data-driven user simulation for automated evaluation of spoken dialog systems. Computer Speech & Language **23**(4), 479–509 (2009)
11. Kreyssig, F., Casanueva, I., Budzianowski, P., Gasic, M.: Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. In: Proc. of SIGdial. pp. 60–69 (2018), https://aclanthology.info/papers/W18-5007/w18-5007
12. Kurata, G., Xiang, B., Zhou, B.: Labeled data generation with encoder-decoder LSTM for semantic slot filling. In: INTERSPEECH 2016. pp. 725–729 (2016). https://doi.org/10.21437/Interspeech.2016-727, https://doi.org/10.21437/Interspeech.2016-727
13. Li, X., Chen, Y.N., Li, L., Gao, J.: End-to-End Task-Completion Neural Dialogue Systems. In: Proc. of IJCNLP (2017), http://arxiv.org/abs/1703.01008
14. Li, X., Lipton, Z.C., Dhingra, B., Li, L., Gao, J., Chen, Y.N.: A user simulator for task-completion dialogues. arXiv preprint arXiv:1612.05688 (2016)
15. Liu, B., Lane, I.: Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In: ASRU Workshop. pp. 482–489. IEEE (2017)
16. Palatucci, M., Pomerleau, D., Hinton, G.E., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., Culotta, A. (eds.) Proc. of NIPS, pp. 1410–1418. Curran Associates, Inc. (2009), http://papers.nips.cc/paper/3650-zero-shot-learning-with-semantic-output-codes.pdf
17. Pietquin, O., Dutoit, T.: A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Trans on Audio, Speech, and Language Processing **14**(2), 589–599 (2006)

18. Roy, N., Pineau, J., Thrun, S.: Spoken dialogue management using probabilistic reasoning. In: Proc. of ACL (2000), http://www.aclweb.org/anthology/P00-1013
19. Schatzmann, J., Stuttle, M.N., Karl, W., Young, S.: Effects of the user model on simulation-based learning of dialogue strategies. In: Proc. of ASRU (2005)
20. Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S.: Agenda-based user simulation for bootstrapping a pomdp dialogue system. In: Proc. of NAACL. pp. 149–152. Association for Computational Linguistics (2007)
21. Schatzmann, J., Thomson, B., Young, S.: Statistical user simulation with a hidden agenda. Proc. of SIGDial **273282**(9) (2007)
22. Schatzmann, J., Weilhammer, K., Stuttle, M., Young, S.: A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. The knowledge engineering review **21**(2), 97–126 (2006)
23. Shah, P., Hakkani-Tür, D., Tür, G., Rastogi, A., Bapna, A., Nayak, N., Heck, L.: Building a conversational agent overnight with dialogue self-play. arXiv preprint arXiv:1801.04871 (2018)
24. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech & Language **21**(2), 393–422 (2007)
25. Young, S., Gašić, M., Thomson, B., Williams, J.D.: Pomdp-based statistical spoken dialog systems: A review. Proc. of the IEEE **101**(5), 1160–1179 (2013)
26. Zhao, T., Eskénazi, M.: Zero-shot dialog generation with cross-domain latent actions. CoRR **abs/1805.04803** (2018), http://arxiv.org/abs/1805.04803