# End-to-End Task-Completion Neural Dialogue Systems

**Xiujun Li**[†] **Yun-Nung Chen**[⋆] **Lihong Li**[†] **Jianfeng Gao**[†] **Asli Celikyilmaz**[†]
[†]Microsoft Research, Redmond, WA, USA
[⋆]National Taiwan University, Taipei, Taiwan
[⋆]y.v.chen@ieee.org
[†]{xiul,lihongli,jfgao,aslicel}@microsoft.com

## Abstract

One of the major drawbacks of modularized task-completion dialogue systems is that each module is trained individually, which presents several challenges. For example, downstream modules are affected by earlier modules, and the performance of the entire system is not robust to the accumulated errors. This paper presents a novel end-to-end learning framework for task-completion dialogue systems to tackle such issues. Our neural dialogue system can directly interact with a structured database to assist users in accessing information and accomplishing certain tasks. The reinforcement learning based dialogue manager offers robust capabilities to handle noises caused by other components of the dialogue system. Our experiments in a movie-ticket booking domain show that our end-to-end system not only outperforms modularized dialogue system baselines for both objective and subjective evaluation, but also is robust to noises as demonstrated by several systematic experiments with different error granularity and rates specific to the language understanding module[1].

## 1 Introduction

In the past decade, goal-oriented dialogue systems have been the most prominent component in today's virtual personal assistants, which allow users to speak naturally in order to accomplish tasks more efficiently. Traditional systems have a rather complex and modularized pipeline, consisting of a language understanding (LU) module, a dialogue manager (DM), and a natural language generation (NLG) component (Rudnicky et al., 1999; Zue et al., 2000; Zue and Glass, 2000).

Recent advances of deep learning have inspired many applications of neural models to dialogue systems. Wen et al. (2017) and Bordes et al. (2017) introduced a network-based end-to-end trainable task-oriented dialogue system, which treated dialogue system learning as the problem of learning a mapping from dialogue histories to system responses, and applied an encoder-decoder model to train the whole system. However, the system is trained in a supervised fashion: not only does it require a lot of training data, but it may also fail to find a good policy robustly due to lack of exploration of dialogue control in the training data. Zhao and Eskenazi (2016) first presented an end-to-end reinforcement learning (RL) approach to dialogue state tracking and policy learning in the DM. This approach is shown to be promising when applied to the task-oriented dialogue problem of guessing the famous person a user thinks of. In the conversation, the agent asks the user a series of *Yes/No* questions to find the correct answer. However, this simplified task may not generalize to practical problems due to the following:

1. **Inflexible question types** — asking request questions is more natural and efficient than *Yes/No* questions. For example, it is more natural and efficient for the system to ask "*Where are you located?*" instead of "*Are you located in Palo Alto?*", when there are a large number of possible values for the location slot.
2. **Poor robustness** — the user answers are too simple to be misunderstood, so the system lacks the robustness against noise in real user utterances.
3. **User requests during dialogues** — in a task-oriented dialogue, user may ask questions for

[1]The source code is available at: https://github/com/MiuLab/TC-Bot.

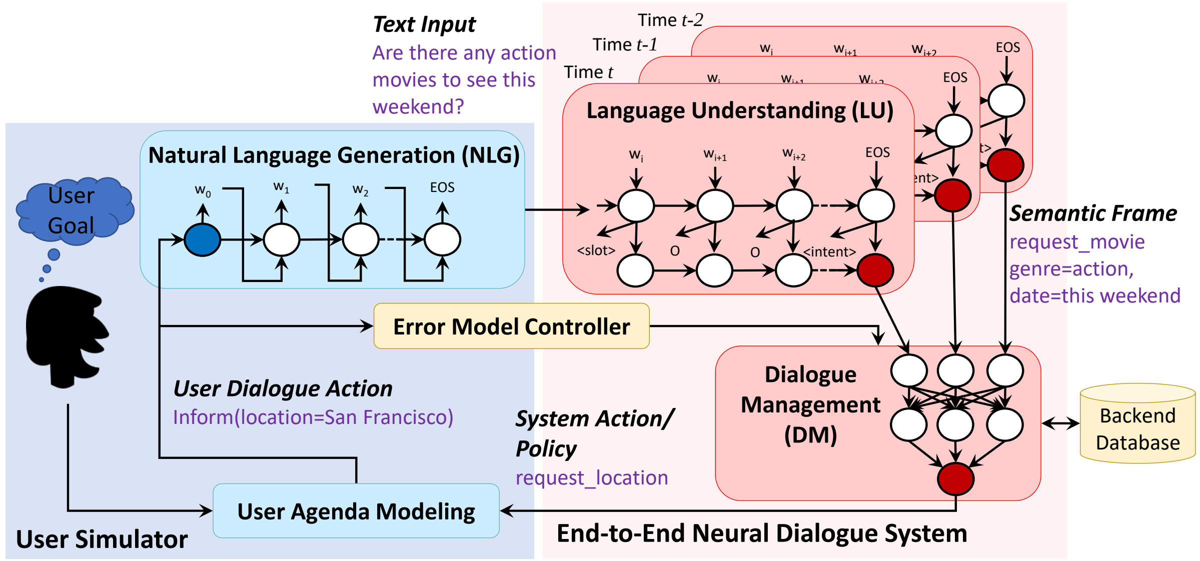



Figure 1: Illustration of the end-to-end neural dialogue system: given user utterances, reinforcement learning is used to train all components in an end-to-end fashion.

selecting the preferred slot values. In a flight-booking example, user might ask "*What flight is available tomorrow?*".

For the second issue, Su et al. (2016) briefly investigated the effect of dialogue action level semantic error rates on the dialogue performance. Lemon and Liu (2007) compared policy transfer properties under different environments, showing that policies trained in high-noise conditions have better transfer properties than those trained in low-noise conditions. Recently, Dhingra et al. (2017) proposed an end-to-end differentiable KB-Infobot to provide the solutions to the first two issues, but the last one remained unsolved.

This paper addresses all three issues above by redefining the targeted system as a task-completion neural dialogue system. Our framework is more practical in that the information can be easily accessed by the user during the conversations, while the final goal of the system is to complete a task, such as movie-ticket booking. This paper is the first attempt of training a real-world task-completion dialogue system in an end-to-end fashion by leveraging supervised learning and reinforcement learning techniques. To further understand the robustness of reinforcement learning based dialogue systems, we conduct extensive experiments and quantitative analysis on a fine-grained level of LU errors, and provide meaningful insights on how the language understanding component impacts the overall performance of the dialogue system.

Our contributions are three-fold:

- **Robustness** — We propose a neural dialogue system with greater robustness by automatically selecting actions based on uncertainty and confusion by reinforcement learning. We also provide the first systematic analysis to investigate the impact of different types of natural language understanding errors on dialogue system performance. We show that slot-level errors have a greater impact on the system performance than intent-level ones, and that slot value replacement degrades the performance most. Our findings shed some light on how to design multi-task natural language understanding models (intent classification and slot labeling) in the dialogue systems with consideration of error control.
- **Flexibility** — The system is the first neural dialogue system that allows user-initiated behaviors during conversations, where the users can interact with the system with higher flexibility that is important in realistic scenarios.
- **Reproducibility** — We demonstrate how to evaluate RL dialogue agents using crowd-sourced task-specific datasets and simulated users in an end-to-end fashion, guaranteeing reproducibility and consistent comparisons of competing methods in an identical setting.

| | | | | | |
|---|---|---|---|---|---|
| **W** | find | action | movies | this | weekend |
| | ↓ | ↓ | ↓ | ↓ | ↓ |
| **S** | O | B-genre | O | B-date | I-date |
| **I** | find_movie | | | | |

Figure 2: An example utterance with annotations of semantic slots in IOB format (S) and intent (I), B-date and I-date denote the date slot.

## 2 Proposed Framework

The proposed framework[2] is illustrated in Figure 1. It includes a user simulator (left part) and a neural dialogue system (right part). In the user simulator, an agenda-based user modeling component based at the dialogue act level is applied to control the conversation exchange conditioned on the generated user goal, to ensure the user behaves in a consistent, goal-oriented manner. An NLG module is used to generate natural language texts corresponding to the user dialogue actions. In a neural dialogue system, an input sentence (recognized utterance or text input) passes through an LU module and becomes a corresponding semantic frame, and an DM, which includes a state tracker and policy learner, is to accumulate the semantics from each utterance, robustly track the dialogue states during the conversation, and generate the next system action.

### 2.1 Neural Dialogue System

**Language Understanding (LU):** A major task of LU is to automatically classify the domain of a user query along with domain specific intents and fill in a set of slots to form a semantic frame. The popular IOB (in-out-begin) format is used for representing the slot tags, as shown in Figure 2.

$$\begin{aligned} \vec{x} &= w_1, ..., w_n, \text{<EOS>} \\ \vec{y} &= s_1, ..., s_n, i_m \end{aligned}$$

where $\vec{x}$ is the input word sequence and $\vec{y}$ contains the associated slots, $s_k$, and the sentence-level intent $i_m$. The LU component is implemented with a single LSTM, which performs intent prediction and slot filling simultaneously (Hakkani-Tür et al., 2016; Chen et al., 2016):

$$\vec{y} = \text{LSTM}(\vec{x}) \,. \tag{1}$$

The LU objective is to maximize the conditional probability of the slots and the intent $\vec{y}$ given the word sequence $\vec{x}$:

$$p(\vec{y} \mid \vec{x}) = \left( \prod_i^n p(s_i \mid w_1, \dots, w_i) \right) p(i_m \mid \vec{y}).$$

The weights of the LSTM model are trained using backpropagation to maximize the conditional likelihood of the training set labels. The predicted tag set is a concatenated set of IOB-format slot tags and intent tags; therefore, this model can be trained using all available dialogue actions and utterance pairs in our labeled dataset in a supervised manner.

**Dialogue Management (DM):** The symbolic LU output is passed to the DM in the dialogue act form (or semantic frame). The classic DM includes two stages, *dialogue state tracking* and *policy learning*.

- **Dialogue state tracking**: Given the LU symbolic output, such as request(moviename; genre=action; date=this weekend), three major functions are performed by the state tracker: a symbolic query is formed to interact with the database to retrieve the available results; the state tracker will be updated based on the available results from the database and the latest user dialogue action; and the state tracker will prepare the state representation $s_t$ for policy learning.
- **Policy learning**: The state representation for the policy learning includes the latest user action (e.g., request(moviename; genre=action; date=this weekend)), the latest agent action (request(location)), the available database results, turn information, and history dialogue turns, etc. Conditioned on the state representation $s_t$ from the state tracker, the policy $\pi$ is to generate the next available system action $a_t$ according to $\pi(s_t)$. Either supervised learning or reinforcement learning can be used to optimize $\pi$. Details about RL-based policy learning can be found in section 3.

Prior work used different implementation approaches summarized below. Dialogue state tracking is the process of constantly updating the state of the dialogue, and Lee (2014) showed that there is a positive correlation between state tracking performance and dialogue performance. Most production systems use manually designed heuristics, often based on rules, to update the dialogue

[2]The source code is available at: `https://github.com/MiuLab/TC-Bot`

states based on the highly confident output from LU. Williams et al. (2013) formalized the tracking problem as a supervised sequence labeling task, where the input is LU outputs and the output is the true slot values, and the state tracker's results can be translated into a dialogue policy. Zhao and Eskenazi (2016) proposed to jointly train the state tracker and the policy learner in order to optimize the system actions more robustly. Instead of explicitly incorporating the state tracking labels, this paper learns the system actions with implicit dialogue states, so that the proposed DM can be more flexible and robust to the noise propagated from the previous components (Su et al., 2016; Liu and Lane, 2017). A rule-based agent is employed to warm-start the system, via supervised learning on labels generated by the rules. The system is then further trained end-to-end with RL, as explained in section 3.

### 2.2 User Simulation

In order to perform end-to-end training for the proposed neural dialogue systems, a user simulator is required to automatically and naturally interact with the dialogue system. In the task-completion dialogue setting, the user simulator first generates a user goal. The agent does not know the user goal, but tries to help the user accomplish it in the course of conversations. Hence, the entire conversation exchange is around this goal implicitly. A user goal generally consists of two parts: *inform_slots* for slot-value pairs that serve as constraints from the user, and *request_slots* for slots whose value the user has no information about, but wants to get the values from the agent during the conversation. The user goals are generated using a set of labeled conversational data.

**User Agenda Modeling:** During the course of a dialogue, the user simulator maintains a compact, stack-like representation called *user agenda* (Schatzmann and Young, 2009), where the user state $s_u$ is factored into an agenda $A$ and a goal $G$. The goal consists of constraints $C$ and request $R$. At each time-step $t$, the user simulator generates the next user action $a_{u,t}$ based on the current state $s_{u,t}$ and the last agent action $a_{m,t-1}$, and then updates the current status $s'_{u,t}$.

**Natural Language Generation (NLG):** Given the user's dialogue actions, the NLG module generates natural language texts. To control the quality of user simulation given limited labeled data, a hybrid approach including a template-based NLG and a model-based NLG is employed, where the model-based NLG is trained on the labeled dataset with a sequence-to-sequence model. It takes dialogue acts as input, and generates sentence sketch with slot placeholders via an LSTM decoder. Then a post-processing scan is performed to replace the slot placeholders with their actual values (Wen et al., 2015). In the LSTM decoder, we apply beam search, which iteratively considers the top $k$ best sub-sentences when generating the next token.

In the hybrid model, if the user dialogue actions can be found in the predefined sentence templates, the template-based NLG is applied; otherwise, the utterance is generated by the model-based NLG. This hybrid approach allows a dialogue system developer to easily improve NLG by providing templates for sentences that the machine-learned model does not handle well.

### 2.3 Error Model Controller

When training or testing a policy based on semantic frames of user actions, an error model (Schatzmann et al., 2007) is introduced to simulate noises from the LU component, and noisy communication between the user and the agent in order to test the model robustness. Here, we introduce different levels of noises in the error model: one type of errors is at the *intent* level, another is at the *slot* level. For each level, there are more fine-grained noises.

**Intent-Level Error:** At the intent level, we categorize all intents into three groups:

- *Group 1*: general *greeting*, *thanks*, *closing*, etc.
- *Group 2*: users may *inform*, to tell the slot values (or constraints) to the agent, for example, inform(moviename='Titanic', starttime='7pm').
- *Group 3*: users may *request* information for specific slots. In a movie-booking scenario, users might ask "request(starttime; moviename='Titanic')".

In the specific task of movie-booking, for instance, there exist multiple *inform* and *request* intents, such as request_starttime, request_moviename, inform_starttime and inform_moviename, etc. Based on the above intent categories, there are three types of intent errors:

- *Random error (I0)*: the random noisy intent from the same category (*within group error*) or other categories (*between group error*).

- *Within-group error (I1)*: the noisy intent is from the same group of the real intent, for example, the real intent is request_theater, but the predicted intent from LU module might be request_moviename.
- *Between-group error (I2)*: the noisy intent is from the different group, for example, a real intent request_moviename might be predicted as the intent inform_moviename.

**Slot-level Error:** At the slot level, there are four error types:

- *Random error* (S0): to simulate the noise that is randomly set to the following three types.
- *Slot deletion* (S1): is to simulate the scenario where the slot is not recognized by the LU component.
- *Incorrect slot value* (S2): is to simulate the scenario where the slot name is correctly recognized, but the slot value is wrong, e.g., wrong word segmentation.
- *Incorrect slot* (S3): is to simulate the scenario where both the slot and its value are incorrectly recognized.

## 3 End-to-End Reinforcement Learning

To learn the interactive policy of our system, we apply reinforcement learning to the DM training in an end-to-end fashion, where each neural network component can be fine tuned. The policy is represented as a deep Q-network (DQN) (Mnih et al., 2015), which takes the state $s_t$ from the state tracker as input, and outputs $Q(s_t, a; \theta)$ for all actions $a$. Two important DQN tricks, target network usage and experience replay are applied, where the experience replay strategy is changed for the dialogue setting.

During training, we use $\epsilon$-greedy exploration and an experience replay buffer with dynamically changing buffer size. At each simulation epoch, we simulate $N$ ($N = 100$) dialogues and add these state transition tuples $(s_t, a_t, r_t, s_{t+1})$ to the experience replay buffer for training. In one simulation epoch, the current DQN will be updated multiple times (depending on the batch size and the current size of experience replay buffer). At the last simulation epoch, the target network will be replaced by the current DQN, the target DQN network is only updated for once in one simulation epoch.

The experience replay strategy is critical for RL training (Schaul et al., 2015). In our buffer update strategy, we accumulate all experience tuples from the simulation and flush the pool till the current RL agent reaches a success rate threshold (i.e., a threshold which is equal to the performance of a rule-based agent), and then use the experience tuples from the current RL agent to re-fill the buffer. The intuition is that the initial performance of the DQN is not strong enough to generate good experience replay tuples, thus we do not flush the experience replay pool till the current RL agent can reach a certain success rate (for example, the success rate of a rule-based agent). In the rest of the training process, at every simulation epoch, we estimate the success rate of the current DQN agent (by running it multiple dialogues on simulated users). If the current DQN agent is better than the target network, the experience replay buffer will be flushed.

## 4 Experiments

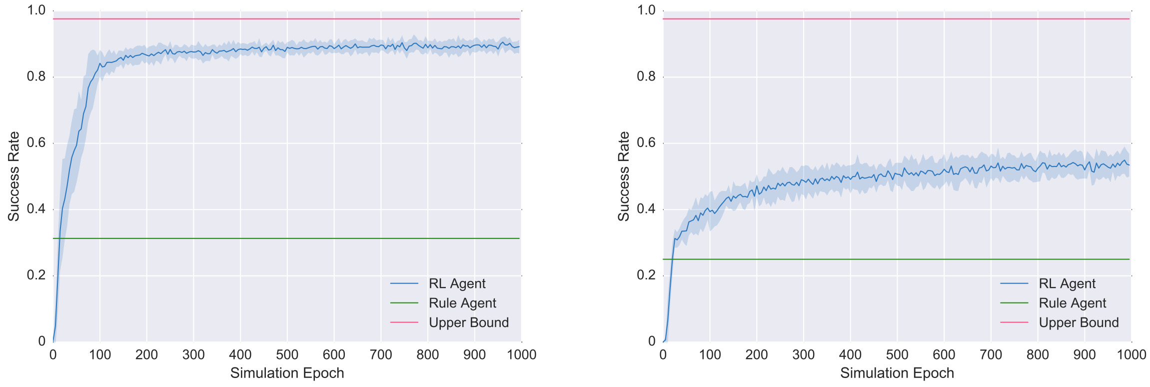
We consider a task-completion dialogue system for helping users book movie tickets. Over the course of conversation, the dialogue system gathers information about the customer's desires and ultimately books the movie tickets. The environment then assesses a binary outcome (success or failure) at the end of the conversation, based on (1) whether a movie is booked, and (2) whether the movie satisfies the users constraints.

**Dataset:** The raw conversational data were collected via Amazon Mechanical Turk, with annotations provided by domain experts. In total, we have labeled 280 dialogues, and the average number of turns per dialogue is approximately 11. The annotated data includes 11 dialogue acts and 29 slots, most of the slots are *informable* slots, which users can use to constrain the search, and some are *requestable* slots, of which users can ask values from the agent. For example, *numberofpeople* cannot be a requestable slot, since arguably user knows how many tickets he or she wants to buy. The detailed annotations can be found in Appendix A.

### 4.1 Simulated User Evaluation

Two sets of experiments are conducted in the DM training, where two input formats are used for training the RL agents:

1. *frame-level semantics*: when training or testing a policy based on semantic frames of user




(a) Frame-level semantics for training

(b) Natural language for end-to-end training

Figure 3: Learning curves for policy training (average of 10 runs). The blue solid lines show the rule agent performance, where we employ to initialize the experience replay buffer pool; the orange dotted line is the optimal upper bound, which is the percentage of reachable user goals.

| | Setting | Intent Error Type | Intent Error Rate | Slot Error Type | Slot Error Rate |
|---|---|---|---|---|---|
| Basic | B1 | | 0.00 | | 0.00 |
| | B2 | 0: random | 0.10 | 0: random | 0.10 |
| | B3 | | 0.20 | | 0.20 |
| Intent | I0 | **0: random** | 0.10 | | |
| | I1 | **1: within group** | 0.10 | | |
| | I2 | **2: between group** | 0.10 | 0: random | 0.05 |
| | I3 | 0: random | **0.00** | | |
| | I4 | 0: random | **0.10** | | |
| | I5 | 0: random | **0.20** | | |
| Slot | S0 | | | **0: random** | 0.10 |
| | S1 | | | **1: deletion** | 0.10 |
| | S2 | | | **2: value** | 0.10 |
| | S3 | 0: random | 0.10 | **3: slot** | 0.10 |
| | S4 | | | 0: random | **0.00** |
| | S5 | | | 0: random | **0.10** |
| | S6 | | | 0: random | **0.20** |

Table 1: Experimental settings with different intent/slot error types described in section 2.3 and different error rates.

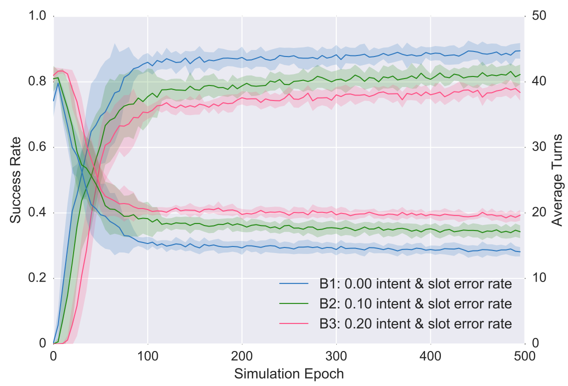
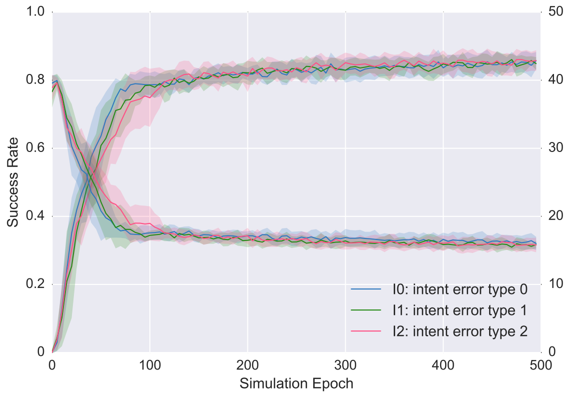



Figure 4: Learning curves for different LU error rates.

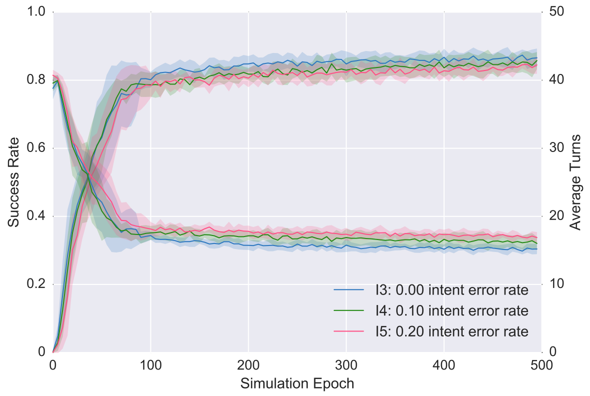
actions, a noise controller described in section 2.3 is used to simulate LU errors and noisy communications between the user and the agent.

2. *natural language*: when training or testing a policy on natural language level, in which LU and NLG may introduce noises. In our experiments, the NLG decoder uses $beam\_size = 3$ to balance speed and performance.

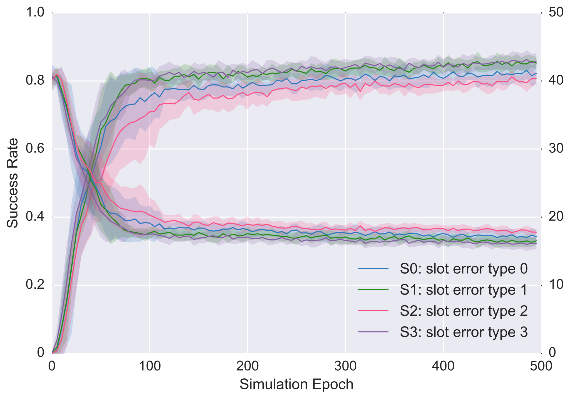
Figure 3(a) shows a learning curve for the dialogue system performance trained with the frame-level information (user semantic frames and system actions), where the number is the average of 10 runs. Figure 3(b) is a learning curve for the system trained at the natural language level. In both settings, the RL agents significantly outperform the rule-based systems, showing the potential of a neural dialogue system that can perform real-world tasks and be improved autonomously through interactions with users. Also, the end-to-end system in Figure 3(b) takes longer for the RL agent to adapt to the noises from LU and NLG, indicating the difficulty of maintaining the system robustness. The consistently increasing trend of our proposed end-to-end system also suggests greater robustness in noisy, real-world scenarios. To further investigate and understand the real impact of the LU component to the robustness of RL agent in the dialogue system, we conduct a series of experiments under different error settings (intent and slot errors from LU) summarized in Table 1, where the learning curves are averaged over 10 runs.



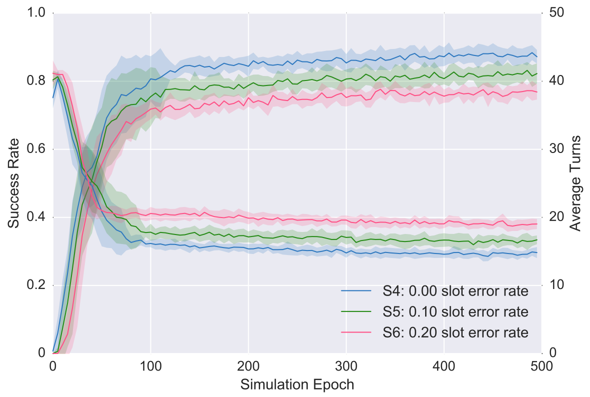

(a) Intent Error Type Analysis




(b) Intent Error Rate Analysis




(c) Slot Error Type Analysis




(d) Slot Error Rate Analysis

Figure 5: Learning curves of the different intent and slot errors in terms of success rate (left axis) and average turns (right axis).

### 4.2 Basic Error Analysis

The group of basic experiments (from B1 to B3) are in the settings that combine the noise from both intent and slot: 1) For both intent and slot, the error types are random, and the error rates are in $\{0.00, 0.10, 0.20\}$. The rule-based agent reports $41\%$, $21\%$, and $12\%$ success rates under $0.00$, $0.10$, and $0.20$ error rates respectively. In contrast, the RL-based agent achieves $90\%$, $79\%$, and $76\%$ success rate under the same error rates, respectively. We compare the performance between two types of agents and find that the RL-based agent has greater robustness and is less sensitive to noisy inputs. Therefore, the following experiments are performed using a RL dialogue agent due to robustness consideration. From Fig. 4, the dialogue agents degrade remarkably when the error rate increases (leading to lower success rates and higher average turns).

### 4.3 Intent Error Analysis

To further understand the impact of intent-level noises to dialogue systems, two experimental groups are performed: the first group (I0–I2) focuses on the difference among all intent error types; the second group (I3–I5) focuses on the impact of intent error rates. Other factors are identical for the two groups, with the random slot error type and a $5\%$ slot error rate.

#### 4.3.1 Intent Error Type

Experiments with the settings of I0–I2 are under the same slot errors and same intent error rate ($10\%$), but with different intent error types: I1 includes the noisy intents from the same categories, I2 includes the noisy intents from different categories, and I0 includes both via random selection. Fig. 5(a) shows the learning curves for all intent error types, where the difference among three curves is insignificant, indicating that the incorrect intents have similar impact no matter what categories they belong to.

### 4.3.2 Intent Error Rate

Experiments with the settings I3–I5 investigate the difference among different intent error rates. When the intent error rate increases, the dialogue agent performs slightly worse, but the difference is subtle. It suggests that the RL-based agent has better robustness to noisy intents. As shown in Fig. 5(a,b), all RL agents can converge to a similar success rate in both intent error type and intent error rate settings.

## 4.4 Slot Error Analysis

We further conducted two groups of experiments to investigate the impact of slot-level noises where other factors are fixed — with the random intent error type and a 10% intent error rate.

### 4.4.1 Slot Error Type

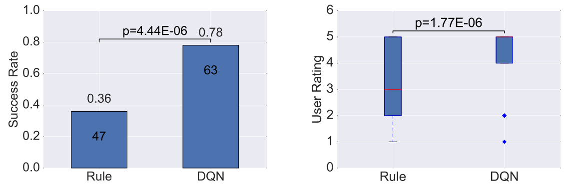
Experiments (S0 – S3) investigate the impact of different slot error types. Corresponding learning curves are given in Fig. 5(c). Among single error types (S1–S3), *incorrect slot value* (S2) performs worst, which means that the slot name is recognized correctly, but a wrong value is extracted with the slot (such as wrong word segmentation); in this case, the agent receives a wrong value for the slot, and eventually books a wrong ticket or fails to book it. The probable reason is that the dialogue agent has difficulty identifying the mistakes, and using the incorrect slot values for the following dialogue actions could significantly degrade the performance. Between *slot deletion* (S1) and *incorrect slot* (S3), the difference is limited, indicating that the RL agent has similar capability of handling these two kinds of slot-level noises.

### 4.4.2 Slot Error Rate

Experiments with the settings from S4 to S6 focus on different slot error rates (0%, 10%, and 20%) and report the results in Fig. 5(d). It is clear from Fig. 5(d) that the dialogue agent performs worse as the slot error rate increases (the curve of the success rate drops and the curve of average turns rises). Comparing with Fig. 5(b), the dialogue system performance is more sensitive to the slot error rate than the intent error rate.

## 4.5 Human Evaluation

We further evaluated the rule-based and DQN agents against real human users recruited from the authors affiliation, where the DQN agent was trained on the simulated user in the frame-level with 5% random slot errors. In each dialogue session, one of the agents was randomly picked to converse with a user, and the user was presented with a predefined user goal sampled from our corpus, and was instructed to converse with the agent to complete the presented task. At the end of each session, the user was asked to give a rating on a scale from 1 (worst) to 5 (best) based on both *naturalness* and *coherence* of the dialogue. We collected a total of 110 dialogue sessions from 8 human users. Figure 6(a) presents the performance of these agents against real users in terms of success rate. Figure 6(b) shows the subjective evaluation in terms of user rating. For all the cases, the RL agent significantly outperforms the rule-based agent for both objective (success rate) and subjective evaluation (user rating).




(a) Success Rate (b) User Rating Distribution

Figure 6: Performance of the rule-based agent versus DQN agent tested with real users: (a) success rate, number of tested dialogues and p-values are indicated on each bar; (b) Distribution of user ratings for two agents (difference in mean is significant with $p < 0.01$).

# 5 Discussion and Future Work

This paper presents an end-to-end learning framework for task-completion neural dialogue systems. Our experiments, both on simulated and real users, show that reinforcement learning systems outperform rule-based agents and have better robustness to allow natural interactions with users in real-world task-completion scenarios. Furthermore, we conduct a series of extensive experiments to understand the impact of natural language understanding errors on the performance of a reinforcement learning based, task-completion neural dialogue system. Our empirical results suggest several interesting findings: 1) slot-level errors have a greater impact than intent-level errors; A possible explanation is related to our dialogue action representation, *intent(slot-value pairs)*. If an intent is predicted wrong, for example, *inform* was predicted incorrectly as *request_ticket*, the dialogue agent can handle this unreliable situation and decide to make confirmation in order to keep the correct information for the following conversation. In contrast, if a slot *moviename* is predicted wrong, or a slot value is not identified correctly, this dialogue turn might directly pass the wrong information to the agent, which might lead the agent to book a wrong ticket. Another reason is that the dialogue agent can still maintain a correct intent based on slot information even though the predicted intent is wrong. In order to verify the hypotheses, further experiments are needed, which we leave as future work. 2) different slot error types have different impacts on the RL agents. 3) RL agents are more robust to certain types of slot-level errors — the agents can learn to double-check or confirm with users, at the cost of slightly longer conversations.

Finally, it should be noted that the experiments in this paper focus on task-completion dialogues. Another type of dialogues known as chit-chats has different optimization goals (Li et al., 2016). It would be interesting to extend our analysis from this paper to chit-chat dialogues to gain useful insights for impacts of LU errors.

## Acknowledgments

We would like to thank Dilek Hakkani-Tür and reviewers for their insightful comments on the paper. Yun-Nung Chen is supported by the Ministry of Science and Technology of Taiwan and MediaTek Inc..

## References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of ICLR*.

Yun-Nung Chen, Dilek Hakanni-Tür, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. 2016. Syntax or semantics? knowledge-guided joint semantic frame parsing. In *Proceedings of the 6th IEEE Workshop on Spoken Language Technology*. pages 348–355.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 484–495.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of Interspeech*. pages 715–719.

Sungjin Lee. 2014. Extrinsic evaluation of dialog state tracking and predictive metrics for dialog policy optimization. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 310.

Oliver Lemon and Xingkun Liu. 2007. Dialogue policy learning for combinations of noise and user simulation: transfer results. In *Proc. SIGdial*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation .

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Proceedings of Interspeech*. pages 2506–2510.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518:529–533.

Alexander I Rudnicky, Eric H Thayer, Paul C Constantinides, Chris Tchou, R Shern, Kevin A Lenzo, Wei Xu, and Alice Oh. 1999. Creating natural dialogs in the carnegie mellon communicator system. In *Eurospeech*.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007. Error simulation for training statistical dialogue systems. In *IEEE Workshop on Automatic Speech Recognition & Understanding*.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing* 17(4):733–747.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv:1511.05952* .

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv:1606.02689* .

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. pages 438–449.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1711–1721.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

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*. pages 1–10.

Victor Zue, Stephanie Seneff, James R Glass, Joseph Polifroni, Christine Pao, Timothy J Hazen, and Lee Hetherington. 2000. JUPITER: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing* 8(1):85–96.

Victor W Zue and James R Glass. 2000. Conversational interfaces: Advances and challenges. *Proceedings of the IEEE* 88(8):1166–1180.

## A Dataset Annotation

Table 2 lists all annotated dialogue acts and slots in detail.

| Annotations | |
|---|---|
| Intent | request, inform, deny, confirm_question, confirm_answer, greeting, closing, not_sure, multiple_choice, thanks, welcome |
| Slot | actor, actress, city, closing, critic_rating, date, description, distanceconstraints, greeting, implicit_value, movie_series, moviename, mpaa_rating, numberofpeople, numberofkids, taskcomplete, other, price, seating, starttime, state, theater, theater_chain, video_format, zip, result, ticket, mc_list |

Table 2: The data annotation.

## B Sample Dialogues

Table 3 shows one success and one failure dialogue examples generated by the rule-based agent and RL agent interacting with user simulator in the movie-booking domain. To be informative, we also explicitly show the user goal at the head of the dialogue, the purpose is to help the user to accomplish this goal and book the right movie tickets, but the agent knows nothing about the user goal.

| | |
|---|---|
| New episode, user goal:<br>{<br>"request_slots": {<br>"ticket": "UNK"<br>},<br>"inform_slots": {<br>"city": "seattle",<br>"numberofpeople": "2",<br>"theater": "regal meridian 16",<br>"starttime": "9:25 pm",<br>"date": "tomorrow",<br>"moviename": "zoolander 2"<br>}<br>} | New episode, user goal:<br>{<br>"request_slots": {<br>"ticket": "UNK"<br>"theater": "UNK"<br>"starttime": "UNK"<br>},<br>"inform_slots": {<br>"numberofpeople": "3",<br>"date": "tomorrow",<br>"moviename": "10 cloverfield lane"<br>}<br>} |
| **Rule-based agent** | **Rule-based Agent** |
| User: Can I get some tickets for zoolander 2?<br>Agent: What movie are you interested in?<br>User: I want to watch zoolander 2.<br>Agent: What time would you like to see it?<br>User: I want to watch at 9:25 pm.<br>Agent: What city you would like?<br>User: I want to watch at seattle.<br>Agent: What date would you like to watch it?<br>User: I want to set it up tomorrow.<br>Agent: Which theater would you like?<br>User: I want to watch at regal meridian 16.<br>Agent: How many tickets do you need?<br>User: I want 2 tickets please!<br>Agent: Great - I was able to purchase 2 tickets for you to see zoolander 2 tomorrow at regal meridian 16 theater in seattle at 9:25 pm.<br>User: Thank you.<br>Agent: Thank you. | User: What is the start time for 10 cloverfield lane?<br>Agent: What movie are you interested in?<br>User: I want to watch at 10 cloverfield lane.<br>Agent: What time would you like to see it?<br>User: What time is it playing tomorrow? I need 3 tickets.<br>Agent: What city you would like?<br>User: I do not care.<br>Agent: What date would you like to watch it?<br>User: I want to set it up tomorrow.<br>Agent: Which theater would you like?<br>User: Which theater is available?<br>Agent: How many tickets do you need?<br>User: I want 3 tickets please!<br>Agent: Okay, I have booked 3 tickets for 10 cloverfield lane tomorrow.<br>User: Thank you.<br>Agent: Thank you. |
| **Success** | **Failure:** Agent does not answer all the questions (starttime, theater) from user. |
| **RL agent** | **RL Agent** |
| User: Can I get tickets for zoolander 2 tomorrow?<br>Agent: Which city would you like?<br>User: I want to watch at seattle.<br>Agent: How many tickets do you need?<br>User: I want 2 tickets please!<br>Agent: 9:25 pm is available.<br>User: I want to watch at regal meridian 16.<br>Agent: Great - I was able to purchase 2 tickets for you to see zoolander 2 tomorrow at regal meridian 16 theater in seattle at 9:25 pm.<br>User: Thank you.<br>Agent: Thank you. | User: Which theater can I book 3 tickets for 10 cloverfield lane?<br>Agent: What time would you like to see it?<br>User: Which theater and start time are available tomorrow?<br>Agent: 11:45am is available.<br>User: Which theater is available?<br>Agent: regal la live stadium 14 is available.<br>User: Could you help me to book the tickets?<br>Agent: Okay - I was able to book 3 tickets for you to see 10 cloverfield lane at regal la live stadium 14 at 11:45am tomorrow.<br>User:Thank you.<br>Agent: Thank you. |
| **Success** | **Success** |

Table 3: Two sample dialogues generated by rule-based and RL agents with user simulator: Left column shows both rule and RL agents succeed; Right column shows that rule-based agent fails, while RL agent succeeds.