

End-to-End Task-Completion Neural Dialogue Systems

Xiujun Li[†] Yun-Nung Chen^{*} Lihong Li[†] Jianfeng Gao[†]

[†]Microsoft Research, Redmond, WA

^{*}National Taiwan University, Taipei, Taiwan

xiul@microsoft.com, y.v.chen@ieee.org

Abstract

This paper presents an end-to-end learning framework for task-completion neural dialogue systems, which leverages supervised and reinforcement learning with various deep-learning models. The system is able to interface with a structured database, and interact with users for assisting them to access information and complete tasks such as booking movie tickets. Our experiments in a movie-ticket booking domain show the proposed system outperforms a modular-based dialogue system and is more robust to noise produced by other components in the system.

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 finish tasks more efficiently. Traditional systems have a rather complex and modular pipelines, containing 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 advance of deep learning has inspired many applications of neural models to dialogue systems. Wen et al. (2016) and Bordes and Weston (2016) 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, thus requires a lot of training data, and may not be able

to explore the unknown space that does not exist in the training data for an optimal and robust policy.

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 a task-oriented system, which is to guess 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 for practical usage due to the following issues: (1) **Inflexible question types** — asking request questions is more natural and efficient compared to 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 user utterances. (3) **User request during dialogues** — in a task-oriented dialogue, user may ask questions for selecting the preferred slot values. In a flight-booking example, user might ask “What flight is available tomorrow?”. Dhingra et al. (2016) proposed an end-to-end differentiable KB-Infobot to provide the solutions to the first two issues, but the last one remained.

This paper addresses all three issues by redefining the targeted system as a task-completion dialogue system to make it more practical, where the information access can be taken by the user during the dialogues, but the final goal of the system is to complete a task, such as movie-ticket booking. This paper is the first attempt of leveraging supervised learning and reinforcement learning techniques for training a real-world task-completion dialogue system in an end-to-end fashion. Our contributions are three-fold:

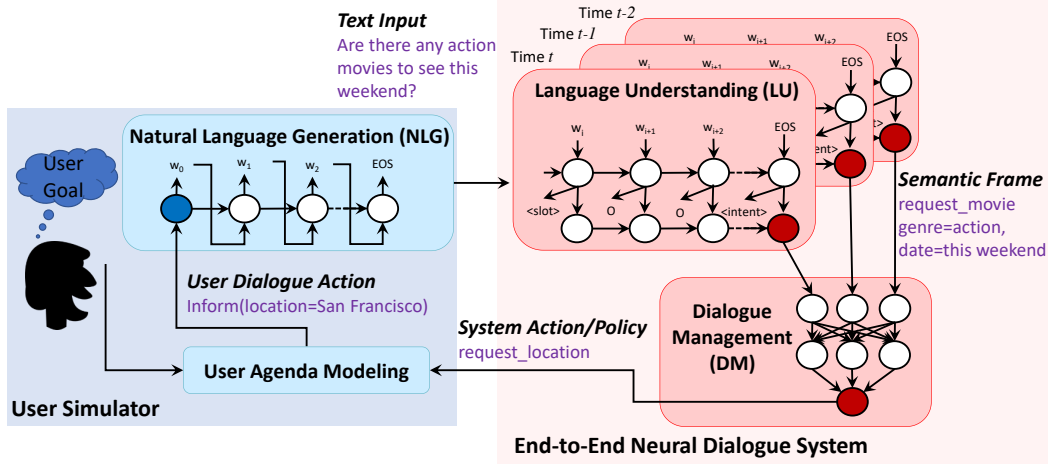


Figure 1: Illustration of the end-to-end **neural dialogue system**: given user utterances, reinforcement learning is used to train all components in an end-to-end fashion.

- **Robustness** — We propose a neural dialogue system with greater robustness by automatically selecting actions based on uncertainty and confusion by reinforcement learning.
- **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 the reproducibility and consistent comparisons of competing methods in an identical setting¹.

2 Proposed Framework

The proposed training framework is illustrated in Figure 1, which includes a user simulator (left part) and a neural dialogue system (right part). In the user simulator, an **agenda-based** user modeling component based on 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 a 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 User Simulation

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 labeled set of conversational data from Li et al. (2016).

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**

¹The code is available at <http://github.com/MiuLab/TC-Bot>.

W	find	action	movies	this	weekend
S	↓	↓	↓	↓	↓
I	O	B-genre	O	B-date	I-date
	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.

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.

2.2 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. 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). 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 a dialogue act form (or semantic frame). The classic DM includes two stages, **dialogue state tracking** and **policy learning**. Given the LU symbolic output, a query is formed to interact with the database to retrieve the available result. The state tracker will be updated based on the available result from the database and the latest user dialogue action. Then the tracker prepares a state representation for the policy learning, including the **latest user action**, **latest agent action**, database results, turn informa-

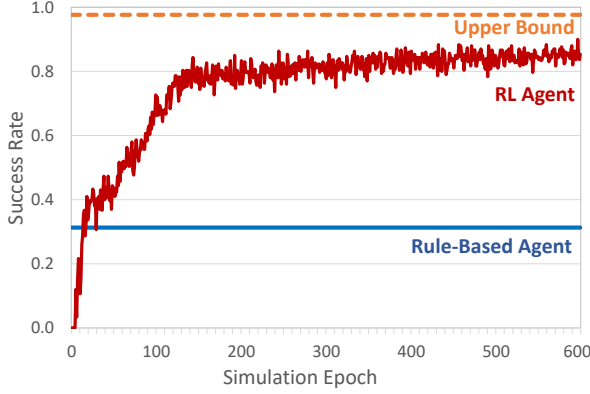
tion, and history dialogue turns, etc. Conditioned on the **state input from the state tracker**, policy learning is to generate the next available system action $\pi(a|s)$. Prior work used different implementation approached summarized below.

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 rule-based heuristics manually defined by dialogue trees to update the dialogue 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 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 noises propagated from the previous components** (Su et al., 2016). A rule-based agent is employed to **warm-start the system** (using supervised learning), which is further trained end-to-end (using RL), as explained below.

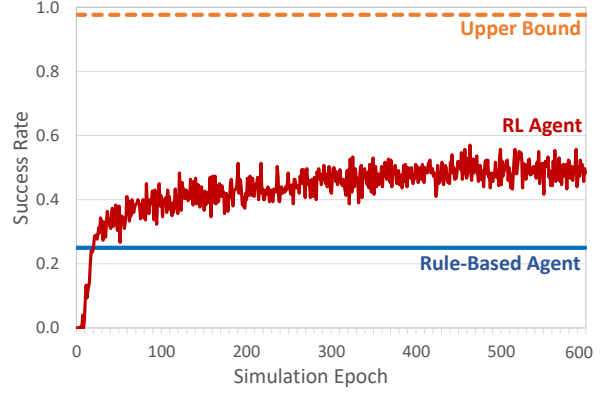
3 End-to-End Reinforcement Learning

To learn the interaction policy of our system, we apply RL to 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 ϵ -greedy exploration and an experience **replay buffer** with dynamically changing buffer size. At each simulation epoch, we simulate N 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



(a) Frame-level semantics for training



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

Figure 3: Learning curves for policy training (average of three 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.

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, 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 good to generate enough 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 times on simulated users). If the current DQN agent is better than the target network, the experience replay buffer will be flushed.

4 Experiments and Discussions

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. 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.

Two sets of experiments are conducted in 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 actions, a noise channel (Schatzmann et al., 2007) 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. More experimental details can be found in the Appendix.

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 three 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 consistent increasing trend of our proposed end-to-end system also suggests greater robustness in noisy, real-world scenarios.

5 Conclusion

This paper presents an end-to-end learning framework for task-completion neural dialogue systems. Our experiments 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.

References

- Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.
- 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 SLT*.
- Bhuwan Dhingra, Lihong Li, Xiuju Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.
- 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.
- Xiuju Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fiedelnd, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan 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. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.
- 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. *arXiv preprint arXiv:1508.01745*.
- 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. *arXiv preprint arXiv:1606.02560*.
- 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.

Table 1: The data annotations.

Annotations	
Dialogue Act	request, inform, confirm_question, confirm_answer, greeting, closing, multiple_choice, thanks, welcome, deny, not_sure
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

A Dataset Annotation

The 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. Table 1 lists all annotated dialogue acts and slots in detail.

B Experimental Detail

B.1 Noise Channel Setup

There are two types of noise channels in the error model: the *intent* level and the *slot* level, where three possible slot errors are introduced:

- *slot deletion*: to simulate the scenario that the slot was not recognized by the NLU;
- *incorrect slot value*: to simulate the scenario that the slot name was recognized correctly, but the slot value was not recognized correctly, e.g., wrong word segmentation;
- *incorrect slot*: to simulate the scenario that both the slot and its value were not recognized correctly.

In our setting, we use the noise channel error rate 0.05 to randomly simulate the slot errors for user turns in order to allow the DM to be more robust to noisy inputs resulted from the machine learned LU or NLG.

B.2 Parameter Setting

In our experiments, we use the same parameter setting. For the NLG decoder we use the *beam_size* = 3 to balance between the speed and performance. For the experience replay buffer update strategy, we set *success_rate_threshold* = 0.30, which is close to the performance of the rule-based agent.

C Sample Dialogues

Table 2 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, but the agent knows nothing about the user goal, its goal is to help the user to accomplish this goal and book the right movie tickets.

Table 2: Two sample dialogues generated by rule-based agent and RL agent with user simulator: Left column shows both rule and RL agents succeed; Right column shows that rule-based agent fails, while RL agent succeeds.

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