Policy Learning for Task-oriented Dialogue Systems via Reinforcement Learning Techniques

Chuandong Yin
ORCID 0000-0002-4682-9733

Submitted in total fulfilment of the requirements of the degree of

Master of Philosophy

School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

May 2018

Abstract

Task-oriented dialogue systems such as Apple Siri and Microsoft Cortana are becoming increasingly popular and are attracting much attention. Task-oriented dialogue systems aim to serve people as virtual personal assistants. For example, they can help users create calendar events or look up the weather through conversations, which improves work efficiency and life convenience. Nevertheless, task-oriented dialogue systems are still in their infant stage and are encountering many problems in *dialogue policy learning* (i.e., learning the next response given a user input and a dialogue context). Most existing work uses supervised learning techniques to learn dialogue policies. However, supervised learning models, especially deep learning models, are usually data-hungry and need a large number of training dialogues. It is difficult to obtain such a large number of training dialogues since intensive labeling efforts are needed to collect training dialogues for a given task domain. Moreover, it is also difficult to measure the quality (correctness) of a training dialogue or the quality of each response in a dialogue, while a supervised learning method needs such information as the training signal to guide policy learning. To overcome these shortcomings, we take a *reinforcement learning* based approach for policy learning in task-oriented dialogue systems. In the reinforcement learning paradigm, a user simulator (i.e., the *environment*) is introduced to mimic various user behaviors and to interact with an *agent* (i.e., a task-oriented dialogue system) that needs to be trained. Via simulated interactions with the user simulator, the agent is able to learn how to make correct responses using a small number of training dialogues and eventually becomes well-trained to serve real users.

We identify two limitations of the reinforcement learning based approach and offer solutions to overcome such limitations in this study. First, existing reinforcement learning based training procedures have not taken into account the context uncertainties of ambiguous user inputs when learning dialogue policies. In task-oriented dialogue systems, user inputs are first transformed to

a series of $\langle entity\_name, entity\_value, confidence \rangle$ tuples by the automatic speech recognition (ASR) and natural language understanding (NLU) components. Here, $entity\_name$ represents an attribute that makes up for a target task (e.g., cuisine types in a restaurant booking task) and $entity\_value$ represents its value (e.g., "Korean"). The $confidence$ field indicates how confident the ASR and NLU components are for recognizing the $entity\_name$ and $entity\_value$ from user input. For ambiguous user input (e.g., due to environment noises or user accents), the confidence might be lower. A low conference value triggers a "confirmation" response of the task-oriented dialogue system, which asks the user to confirm whether the recognized entity name and value are the intended entity name and value. Existing work uses a fixed confidence threshold to trigger confirmation responses. However, the confidence threshold should vary from person to person. For users with accents, even if the entity values are correctly recognized, the confidence is generally lower than those without accents. If a universal confidence threshold is used, it may lead to many rounds of unnecessary confirmations and lengthy dialogues, which will impinge the user experience. To address this issue, we propose to learn a dynamic threshold based on the dialogue context. But learning a dynamic threshold is very challenging because the response (action) space is too large (i.e., each different response of the task-oriented dialogue system may require a different confidence threshold). Finding an optimal dynamic threshold that suits the entire response space needs a large number of simulation steps. As a result, it is difficult for reinforcement learning models to fully explore the response space. We therefore propose a *parameterized auxiliary asynchronous advantage actor-critic* (PA4C) model to solve this problem. PA4C utilizes the action parameterization technique to reduce the size of the response space and introduces auxiliary tasks to efficiently explore responses, which is beneficial to learn the dynamic threshold and to improve task completion rates. We evaluate PA4C on a calendar event creation task, and the experimental results show that PA4C outperforms the state-of-the-art baselines by over 10% in completion rate, dialogue length, and episode reward.

Second, existing task-oriented dialogue systems such as Apple Siri can only handle short dialogues that can be completed in a few dialogue turns, such as looking up the weather. As the dialogue gets longer, learning the optimal policy in each turn becomes increasingly difficult. This is because we can only obtain a global reward at the end of a training dialogue to indicate whether a task has been completed or not. There are no local rewards to evaluate the quality of the re-

sponse in each intermediate dialogue turn. Propagating the global reward back to each dialogue turn to optimize the turn-by-turn policy is challenging. Such a problem is called the *sparse rewards* problem in reinforcement learning and may lead reinforcement learning models into local optima. To address this issue, we propose a new reinforcement learning model named PGGAN to provide local rewards for each dialogue turn by incorporating *policy gradient* (PG) and *generative adversarial networks* (GANs). In particular, we train a *discriminator* to evaluate the similarity between a response produced by the dialogue agent and a human response. This similarity score is then used as the local reward to optimize the dialogue agent (i.e., *generator*) in each intermediate dialogue turn. In this way, the sparse rewards problem can be solved and the dialogue agent can therefore handle long dialogues. We evaluate PGGAN on a public dialogue dataset, and the experimental results show that PGGAN significantly outperforms the state-of-the-art baselines by up to 25% in completion rate.

This page is intentionally left blank.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the MPhil,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 50,000 words in length, exclusive of tables, figures, bibliographies and appendices.

Chuandong Yin, May 2018

This page is intentionally left blank.

Acknowledgements

First of all, I would like to express my deepest gratitude to my principal supervisor Prof. Rui Zhang for his patience, talent, guidance, and continuous support during my MPhil study. His knowledge, skills, and enthusiasm in research have deeply influenced and encouraged me. Without his encouragement, this thesis would not have been possible.

I am deeply grateful to my co-supervisor Dr. Jianzhong Qi. He provides me invaluable discussions and his insightful feedback to my research. He is very supportive and patient. His strict self-discipline and high standard for himself always inspire me to work hard.

I also wish to sincerely thank my advisory committee members: Dr. Rachelle Bosua and Dr. Antonette Mendoza. They keep watching my progress and give me generous support during my MPhil study. Without their insightful feedback and constructive comments, my progress would not have been that smooth.

Then, I would like to thank The University of Melbourne and School of Computing and Information Systems for providing scholarships and SummerTech studentship to financially support my MPhil study. I would also like to thank Google Australia for financially supporting my visit and conference travels.

Last but not least, I would like to thank all my fellow PhD students with whom I share an office or work in various occasions for their support in research, life, and all the pleasant memories, including Zeyi Wen, Jing Huang, Saad Aljubayrin, Jiazhen He, Gitansh Khirbat, Yichen Li, Yuan Li, Xiaojie Wang, Chenxu Zhao, Yiqing Zhang, Wenkai Jiang, Yimeng Dai, Xiaojie Lin, Weihao Chen, Jiayuan He, Aili Shen, Zhen Wang, Ang Li, Daocang Chen, Yunxiang Zhao.

This page is intentionally left blank.

Preface

A paper out of the work presented in Chapter 4 has been accepted and will appear in *The 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*. The work presented in Chapter 5 has been submitted to *IEEE International Conference on Data Mining 2018*. I declare that I am the primary author and have contributed $> 50\%$ in the following papers.

1. Chuandong Yin, Rui Zhang, Jianzhong Qi, Yu Sun, and Tenglun Tan. Context-Uncertainty-Aware Chatbot Action Selection via Parameterized Auxiliary Reinforcement Learning. In *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD), 2018. (Core Ranking [1]: A)

2. Chuandong Yin, Yu Sun, Jianzhong Qi, and Rui Zhang. PGGAN: An Adversarial Approach to Task-Oriented Dialog Policy Learning. Submitted to *the Association for the Advancement of Artificial Intelligence* (AAAI), 2018. (Core Ranking [2]: A*)

[1] http://portal.core.edu.au/conf-ranks/?search=PAKDD&by=all&source=CORE2018&sort=atitle&page=1
[2] http://portal.core.edu.au/conf-ranks/?search=AAAI&by=all&source=CORE2018&sort=atitle&page=1

This page is intentionally left blank.

To my parents and my wife, for their unconditional love.

This page is intentionally left blank.

Contents

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Task-oriented dialogue systems such as Apple Siri, Microsoft Cortana, and Google Now, are gaining in great popularity. The goal of these systems is to serve users as their personal assistants and to provide intelligent service (e.g., booking restaurants and creating calendar events). They attract a large number of studies on various aspects of task-oriented dialogue systems, including automatic speech recognition (ASR) [31, 23, 103], natural language understanding (NLU) [20, 53, 1], dialogue state tracking (DST) [93, 27, 40, 95], dialogue policy learning [42, 96, 102], natural language generation (NLG) [90, 69, 68], and more. In this thesis, we focus on *dialogue policy learning*. The goal of dialogue policy learning is to produce a proper response given a dialogue context and a user utterance. Dialogue policy learning can be defined as $\pi(response|context, utterance)$, where π is the *policy* to learn. Levin et al. [42] model dialogue policy learning as a *Markov decision process* (MDP). Williams and Young [96] further model the problem as a *partially observable Markov decision process* (POMDP). Recently, Rojas-Barahona et al. [64] jointly train NLU, DST, dialogue policy learning, and NLG in an end-to-end manner using supervised learning methods. Dhingra et al. [19] further apply reinforcement learning techniques to dialogue policy learning so as to reduce the number of training dialogues and to improve the task completion rate.

We study dialogue policy learning for task-oriented dialogue systems by reinforcement learning and aim to overcome the problems in existing reinforcement learning models. Compared with existing work, our work has the following main novelties: i) Most existing studies do not take into account the context uncertainty of ambiguous user utterances in dialogue policy learning. In task-oriented dialogue systems, ASR and NLU components give scores on the *confidence* of user utterances. A low confidence score means that dialogue systems may not comprehend user utterances and need to confirm with users. A common approach used by existing work is to confirm

with users until the confidence reaches an empirically pre-defined threshold. In real applications, confidence thresholds vary from person to person due to different language fluency. For example, the confidence scores of users with accents are generally lower than those without. In this case, the confidence threshold should be lower. Otherwise, dialogue systems may confirm with the users in each dialogue turn, which may discourage the use of dialogue systems. We address this issue and build a context-uncertainty-aware dialogue system that can learn a dynamic threshold for different people. ii) Existing task-oriented dialogue systems (e.g., Apple Siri and Google Now) mainly focus on handling simple tasks that can be completed with a short dialogue such as looking up the weather. They are very limited in handling long dialogues that may need multiple dialogue turns to complete, such as making restaurant reservations. As the dialogue gets longer, learning the optimal policy in each turn becomes increasingly difficult. This is because the dialogue system has to memorize user intents in historical turns. However, users may change their intents in the subsequent turns, making it difficult for the dialogue system to correctly identify these changes. We propose to make use of *generative adversarial nets* (GAN) to enhance the memory of reinforcement learning models, which can make dialogue systems more robust when handling long dialogues.

For our first task, (i.e., to handle ambiguous user utterances), we propose a *context-uncertainty-aware dialogue policy* and a novel reinforcement learning model named *parameterized auxiliary asynchronous advantage actor-critic* (PA4C) for training. PA4C overcomes two main problems in context-uncertainty-aware policy learning. The first is the problem of large action spaces. The output of traditional reinforcement learning models is usually a one-hot vector to indicate the action to be selected. Compared with traditional reinforcement learning tasks, the action space is much larger in task-oriented dialogue systems because each action consists of two parts: a function name (i.e., type of actions) and its parameters (i.e., slots). For example, an action *request(time)* contains a function *request* and a parameter *time*. Traditional reinforcement learning models simply list all combinations of available functions and parameters, e.g., {*request(time)*, *request(location)*, *confirm(time)*, *confirm(location)*, ... }. Suppose that there are M functions and N parameters, the number of actions in traditional reinforcement learning models is $M \times N$. This large quadratic action space makes it difficult (i.e., requires more training data and longer training time) to learn the optimal action given a context and user input. To overcome this problem, we propose the

action parameterization technique to reduce the size of action space. Action parameterization allows reinforcement learning models to learn the optimal function and the optimal parameter in two separate channels. Thus, the action space can be reduced from quadratic (i.e., $M \times N$) to linear (i.e., $M + N$). Furthermore, PA4C also tackles the problem of zero-reward dialogue states. In task-oriented dialogue systems, only a few states can provide useful rewards to guide the action exploration. This problem may cause that reinforcement learning models fail to find the optimal action due to insufficient valuable states. Inspired by Jaderberg et al. [35], we propose to use auxiliary tasks to distinguish large-reward states and zero-reward states, making reinforcement learning models more sensitive to capture valuable states because of the explicit attention on the changes of rewards. In particular, we design i) a *reward prediction* task to predict the reward of dialogue states, and ii) a *value function replay* task to estimate the expected state value. In this way, the performance of reinforcement learning models can be significantly improved since both short-term rewards and long-term returns are taken into account.

For the second task (i.e., to handle long dialogues), we propose an *adversarial dialogue policy* and a novel model named PGGAN to train it by systematically integrating *policy gradient* (PG) with GAN. The proposed adversarial dialogue policy trained with PGGAN can solve the *sparse reward* problem in handling long dialogues. We use an example of making restaurant reservations to explain this problem. Suppose that a user is using a task-oriented dialogue system to book a restaurant. The dialogue system starts with requesting the details of user preferences (e.g., the preferred type of cuisine or price range) from the user and then recommends the best restaurant to the user. The user can provide her preferences to the system piece by piece and can determine whether or not to accept the recommended restaurants. At the end of the dialogue, the user gives the system a global positive or negative rating (i.e., reward) as a feedback of the dialogue quality. The dialogue system can get improved based on the given ratings. The dialogue, however, may be lengthy (e.g., over 20 dialogue turns) and only a global reward is given at the end. There are no local rewards on the response quality in each intermediate dialogue turn. This causes the received rewards to become sparse, making it difficult for reinforcement learning models to search for the optimal action and to complete user tasks. To overcome this problem, we propose to use GAN to provide local rewards for intermediate dialogue turns. Specifically, we train a *discriminator* to estimate how much a response generated by a reinforcement learning agent (i.e., *generator*) in

each dialogue turn resembles a human response, and use it as a local reward signal to train the agent. To obtain a larger local reward, the agent needs to generate responses in a human way to confuse the discriminator. In contrast, to provide more accurate local rewards, the discriminator needs to differentiate generated responses from human responses as accurate as possible. Finally, when the discriminator cannot differentiate the responses generated by the agent, it means that the agent can respond like a human. Further, to avoid overly optimizing the agent to obtain large local rewards in dialogue turn t but ignoring global rewards, we propose to perform Monte Carlo simulation for PGGAN from turn $t + 1$ to T, where T is the last dialogue turn. Monte Carlo simulation helps PGGAN take into account both local rewards and global rewards, making the model more reliable when generating responses. In this way, the sparse reward problem can be solved by our adversarial dialogue policy.

1.1 Summary of Contributions

The contributions of this thesis are summarized as follows.

For the task of handling ambiguous user utterances:

- To the best of our knowledge, we are the first to propose a context-uncertainty-aware policy for task-oriented dialogue systems that can be self-adaptive to the uncertainty of ambiguous user utterances and dialogue contexts. We also propose a novel reinforcement learning model named PA4C to train such a dialogue policy.

- To overcome the quadratic action space problem in policy learning via reinforcement learning, we propose the action parameterization technique that learns the functions and slots in two separate channels.

- We further propose two auxiliary networks to guide our model to pay extra attention to valuable states, which is more robust to both short-term immediate rewards and long-term expected returns.

- We perform experiments to verify the effectiveness of the proposed techniques. The results show that our model outperforms the state-of-the-art reinforcement learning models in terms of success rate, dialogue length, and episode reward.

For the task of handling long dialogues:

- We build an adversarial dialogue policy that can handle long dialogues that may take more than 20 turns to complete. In particular, we propose a novel reinforcement learning model named PGGAN by systematically integrating policy gradient with GAN to train such a policy.
- We train a discriminator to differentiate generated responses from human responses and to provide additional turn-based rewards for reinforcement learning models, which can overcome the sparse reward problem in dialogue policy learning.
- When generating responses in dialogue turn t, we take into account both the local reward for the current turn and the global reward for the entire dialogue through *Monte Carlo* simulation from turn $t+1$ to turn T. This makes the generated responses much more reliable.
- We evaluate the proposed PGGAN model by training a task-oriented dialogue system to make restaurant reservations using the adversarial dialogue policy. The experimental results show that PGGAN can significantly reduce the number of training dialogues and consistently improve the task completion rate compared with the baseline models.

1.2 Thesis Structure

The rest of this thesis is organized as follows:

- In Chapter 2, we review related work on task-oriented dialogue policy learning. We discuss different approaches to build task-oriented dialogue systems, such as rule based, retrieval based, supervised learning based, and reinforcement learning based models. We give a brief overview of how to construct a user simulator to simulate real user behaviours when using task-oriented dialogue systems.
- In Chapter 3, we discuss preliminary techniques used in this thesis. Particularly, we focus on reinforcement learning and generative adversarial nets for task-oriented dialogue policy learning. The PA4C model and PGGAN model proposed in the thesis are based on them.

- In Chapter 4, we focus on building a context-uncertainty-aware dialogue policy to handle ambiguous user utterances. First, we give an explanation of the need to build a context-uncertainty-aware dialogue policy. We then explore the problems of large action spaces and zero-reward states when training such a dialogue system using reinforcement learning techniques. Next, we propose a PA4C model to address these two problems. We conduct experiments to validate the effectiveness of the PA4C model and the performance of the PA4C model.

- In Chapter 5, we focus on solving the sparse reward problem in handling long dialogues by an adversarial dialogue policy. We first demonstrate the problem and discuss the drawbacks of existing techniques. Then, we present our proposed PGGAN model that systematically integrates policy gradient, generative adversarial nets, and Mote-Carlo simulation. After that, we perform experiments to validate the effectiveness of the PGGAN model and analyze the reasons why PGGAN outperforms the state-of-the-art models.

- In Chapter 6, we conclude this thesis and discuss future work.

Chapter 2

Literature Review

As mentioned earlier, a task-oriented dialogue policy can be defined as $\pi(response|context, utterance)$. The goal of dialogue policy learning is to produce a proper response given a certain dialogue context and a user utterance. Various types of approaches are used by existing studies to obtain the policy π (i.e., to produce proper responses), including rule based, retrieval based, supervised learning based, and reinforcement learning based approaches. In this chapter, we review related work for context-uncertainty-aware dialogue policies and adversarial dialogue policies in Section 2.1 and Section 2.2, respectively. We also briefly discuss user simulator in Section 2.3 and summarize this chapter in Section 2.4.

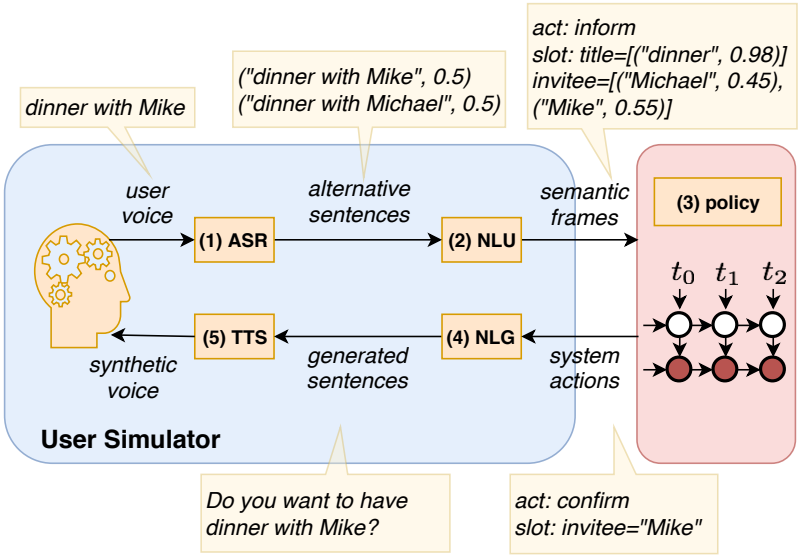


Figure 2.1: A typical framework for task-oriented dialogue system. This framework consists of several independent components: (1) automatic speech recognition (ASR), (2) natural language understanding (NLU), (3) dialogue policy learning, (4) natural language generation (NLG), and (5) text-to-speech (TTS).

2.1 Context-uncertainty-aware Dialogue Policies

In this section, we review related studies for context-uncertainty-aware dialogue policies in handling ambiguous user inputs. We first introduce DST in Section 2.1.1. Then we discuss supervised learning based and reinforcement learning based approaches in Section 2.1.2 and Section 2.1.3, respectively.

2.1.1 Dialogue State Tracking

Task-oriented dialogue systems aim to help users complete a goal (task) via interacting with users in natural language. As the conversation goes on, a dialogue system needs to maintain a representation of the dialogue state. This process is called *dialogue state tracking* (DST) [86]. For example, regarding the task of making restaurant reservations, the dialogue state may contain the user preferences of restaurants, such as the type of cuisine, location, price range, etc. It is not an easy task to track dialogue states since the automatic speech recognition (ASR) and natural language understanding (NLU) components are not perfect, which will produce noises. As shown in Figure 2.1, these noises will propagate throughout the entire framework, causing that dialogue systems may distort or misunderstand user intents. However, DST plays a determinant role in task-oriented dialogue systems. If dialogue states are not accurately tracked, the dialogue system may respond the user with incorrect responses, causing the failure of the conversation. Unfortunately, most of existing commercial task-oriented dialogue systems still use handcrafted heuristic rules to track dialogue states [30, 84]. They only keep the NLU output with the largest confidence score, discarding the other possible results. To overcome this problem, Thomson and Young [78] propose to use statistical methods to calculate the score for each possible result given a dialogue state (context).

However, there is no standard to validate the effectiveness of these statistical methods because they are built on different domains. Williams et al. [93] propose *Dialog State Tracking Challenge 2* (DSTC 2) to address this issue and provide a variety of corpus containing 15K human-computer dialogues in a standard format. Besides, they also propose 11 metrics to evaluate the quality of DST. Typically, DST assumes that ASR and NLU components can output semantic frames in the format of $[(a, s, v, p)]$ pairs, where a denotes the dialogue act, s denotes the slot (i.e., entity name),

v denotes the value of s, and p is the possibility that the value of slot s is v. Table 2.1 shows several specific instances of semantic frames.

Table 2.1: Example semantic frames.

Semantic Frame
(*inform*, people, "Mike", 0.6)
(*inform*, people, "Michael", 0.3)
(*inform*, people, "Mark", 0.1)
(*inform*, location, "Mike", 0.1)
(*inform*, time, "Mark", 0.1)
(*confirm*, people, "Mark", 0.1)
(*confirm*, people, "Mike", 0.5)

DSTC 2 formulates DST as a sequential labeling problem and attracts a number of researchers to study it [94, 93]. In past, researchers preferred to use the generative Bayesian network models for state tracking [60, 78]. Until Hinton et al. [32] proposed deep brief network (DBN) and achieved a notable improvement in speech recognition [31], researchers began to pay more attention to discriminative models. Inspired by the work of Hinton et al. [32], Henderson et al. [27] first used deep neural networks to train discriminative models for DSTC, which significantly improved the accuracy compared with traditional generative models. Unlike traditional machine learning models, deep neural networks can learn much more complex feature representations of dialogue states with a little human labour. Their model can output a sequence of probability distributions over the possible system actions with tied weights and sliding windows. Gradient ascent techniques [12] can be directly applied to maximize the log-likelihood of actions. The derivatives of the optimization objectives with respect to all the model parameters can be computed using back propagation via the chain rule. In particular, they use mini-batch stochastic gradient descent (SGD) to optimize the model parameters in the direction of minimizing the negative log-likelihood [12]. The optimization process will cease if the model performance stops to improve or reaches the pre-defined maximum iteration step. More importantly, such an approach can be easily adapted to new domains without changing the learning objective or model structures. Their model can retain a good performance in spite of dealing with unseen dialogues (e.g., the "test4" dataset provided by DSTC [93]). Their experimental results show that deep neural network is able to learn the hidden representation of dialogues in a semantic manner, which can significantly outperform traditional

supervised learning models.

Henderson et al. [28] make a further progress by utilizing recurrent neural networks (RNNs) to solve the problem of DST. Compared to the work of Henderson et al. [27] which is based on the MDP formulation, Henderson et al. [28] use the POMDP formulation to memorize the historical dialogue states. By combining RNN and POMDP, their model is able to handle unseen dialogue states with a little feature engineering [56]. To represent historical dialogue contexts, conventional models have to pre-process the input with a sliding window with a fixed size [93]. Therefore, the model performance is often influenced by the choice of window size, which is very limited in handling long-term dialogue history. In contrast, RNN can fully make use of the dynamic structure of RNN, making it possible to handle long sequences step by step. It can learn the internal correlation of dialogue contexts regarding both the short-term and the long-term history. Particularly, Henderson et al. [28] feed the current user utterances and the last dialogue response as the input of the most recent dialogue turn. Meanwhile, the internal memory of RNN will be updated and then used to compute the action in the next dialogue turn. To better represent the user input, they extract n-grams from user utterances as features. The n-gram features can be obtained from the ASR candidate list, including the hypothesis of unigram, bigram, and trigram features. Then, a final feature vector is formed by aggregating these n-grams according to their probabilities. These augmented features can increase the robustness of RNN when handling unseen dialogue states. However, deep neural networks are usually suffering from the cold start problem, the problem will become increasingly serious after introducing RNN because of more parameters to learn. To overcome this problem, Henderson et al. [28] also apply denoising autoencoder (DA) [83] to warm up the RNN model. This is because DA can learn the underlying representations in an unsupervised manner and can inject the learned knowledge into the neural units in RNN. In this way, their model dramatically outperforms baselines.

DST is a very important component in task-oriented dialogue systems because subsequent components such as dialogue policy learning and NLG heavily rely on the result of DST. But our work will no focus on studying DST since DST does not aim to make the decision on the next dialogue response. Therefore, in this thesis, we only use DST as the dependency of training context-uncertainty-aware dialogue policies.

2.1.2 Supervised Learning Based Approaches

Conventionally, supervised learning techniques are widely used for dialogue policy learning [99, 98, 24]. The intuition behind this approach is to train a policy model that outputs the next system response given a user utterance and a dialogue context.

Yan et al. [99] build a task-oriented dialogue system to help users search products when shopping online. Their system consists of multiple components, such as query understanding, DST, dialogue policy learning, and product knowledge base. In terms of dialogue policy learning, they divided the actions in their dialogue systems into the following categories: (1) *recommendation* to recommend products to users, (2) *comparison* to compare two products mentioned by users, (3) *opinion summary* to user opinions about the target products, (4) *question answering* to request the details of the item from the user, (5) *proactive questioning* to detect the intents of products, and (6) *chit-chat* to chat with users. To build the dialogue policy, they follow the work of Yih et al. [101] and train a supervised learning model on millions of search log data covering 11M products and 1,080 categories from real users. Specifically, they first parse search logs into a series of $\langle s_t, a_t \rangle$ pairs, where s_t denotes the user input, a_t denotes the system action, and t denotes the dialogue turn. Then user input s_t is fed into the DST component, which can obtain a dialogue state $\mathcal{H}_t$. After this, the policy model can be optimized to output a_t with $\mathcal{H}_t$ as the input. They further deploy their task-oriented dialogue system to an E-commerce platform, where millions of real customers can use it for online shopping.

Furthermore, Williams et al. [98] propose a *hybrid code network* (HCN) to train dialogue policies in the supervised fashion. As mentioned before, most existing task-oriented dialogue systems are composed of several independents modules, such as NLU, policy learning, and NLG. The dependencies among these components are especially complicated, making it difficult to define dialogue states and maintain dialogue contexts. However, the dialogue states recorded by the state tracker heavily affect the performance of policy learning and it is challenging to find the internal correlation among these components in an easy way. Some existing studies attempt to alleviate this problem with end-to-end training [97, 19]. They straightforwardly use RNN to generate responses token by token because RNN is able to learn a latent representation of dialogue states without explicitly labeling states. Nevertheless, a large pitfall in these end-to-end models is that it is hard to inject domain knowledge into the generated responses, such as sorting the results of

database queries or updating the value of entities in the database [11, 64]. It may take a large number of extra dialogues to train a model to master these simple operations. Unfortunately, such programmed constraints are nontrivial in real scenarios. For example, some task-oriented dialogue systems for banks usually require user authentication before doing real tasks. Therefore, they propose a novel model named HCN to overcome these problems. HCN allow developers to customize API actions (searching or updating a back-end database, or calling customized functions) to inject domain knowledge. Compared to existing task-oriented dialogue systems, HCN provide much more freedom for developers to manipulate the trained dialogue systems with only a little human effort. Their experimental results show that HCN can outperform baseline models with a significantly smaller number of training dialogues. To demonstrate the effectiveness of the HCN model, they build a dialogue system application that can be used by real customers and can achieve an amazing result.

Although these studies make great achievements, they still have imperfections. For example, when handling ambiguous user inputs, they only select the most confident ASR result and discard the other possible candidates, which may produce biases. Besides, they use supervised learning techniques to train the dialogue policy. However, supervised learning models cannot incorporate user feedbacks, making the dialogue system unable to adapt to user preferences. In this thesis, we address these two issues and propose corresponding solutions to overcome them in Chapter 4.

2.1.3 Reinforcement Learning Based Approaches

Since Mnih et al. [58] successfully used deep Q-network [25] to play Atari games and outperformed the best human players, reinforcement learning techniques have attracted more and more attention in dialogue policy learning [41, 64, 18, 43]. Reinforcement learning models can interact with a user simulator to explore more responses uncovered by the dataset, which can significantly reduce the human labour in data collection. Furthermore, compared with supervised learning approaches, reinforcement learning can incorporate the user feedback into the training process by means of rewards, making dialogue systems to be adaptive to user preferences.

As shown in Figure 2.1 earlier, a dialogue system consists of several individual components. However, these components are trained separately, leading the noises in previous components to propagate in the subsequent components. This will make the learned dialogue policy unreliable.

Rojas-Barahona et al. [64] address this problem and propose an end-to-end trainable model to jointly train these components. Their model includes the following components: (1) an *intent network* to encode a sequence of word tokens w_0^t, w_1^t, ..., w_N^t into a vector $\mathbf{z}_t$ in each dialogue turn t with a sequence-to-sequence learning model. (2) a *belief tracker* to track dialogue states and map natural language sentences into the format of slot-value pairs so as to query a back-end database [93]. They use weight tying techniques in the belief tracker to reduce the number of training dialogues and to simplify the deployment of dialogue systems in future. (3) a *policy network* to choose the next action according to the current dialogue state. It accepts the result of the intent network, brief tracker, and the database states as the input, and outputs a one-hot vector indicating the selected action. (4) a *generation network* to map the one-hot vector (i.e., the selected action) into a natural language sentence to respond users [90]. This network will output a probability distribution over the possible tokens, and then we can sample a series of tokens from this distribution to generate a sentence. The generated sentence includes many generic tokens (e.g., $\{$s.food$\}$) that can be substituted with specific entities (e.g., "Spanish" or "Korean"). Their model is overall end-to-end trainable, but in order to keep the maintainability of the whole system, these components are still modularly connected. Rojas-Barahona et al. [64] treat task-oriented dialogue system training process as a sequence-to-sequence mapping problem [53] with incorporating dialogue contexts and search results of database queries. So as to generalize to unseen tokens, Rojas-Barahona et al. [64] use delexicalization [29] in the natural language generation components. Such an approach makes the model more flexible and can also reduce the number of training dialogues. Their model can significantly outperform the baseline models when training with only a few hundred dialogues. Because of being trained in an end-to-end manner, the learned dialogue policy is more robust to the noises produced in other components and therefore can achieve a much higher task completion rate. Furthermore, in order to simplify the process of collecting dialogue data, Rojas-Barahona et al. [64] put forward a novel Wizard-of-Oz paradigm that two people mimic the behaviours of a dialogue system and a user via crow-sourcing [36]. This approach can significantly reduce human labour and improve the efficiency of data collection.

Cuayahuitl et al. [18] continue to optimize task-oriented dialogue systems that can handle multi-domain dialogues. They also propose a new reinforcement learning model that can scale up to multiple domains. Conventional deep reinforcement learning models, such as deep Q-Networks

[58], usually encounters a scalability problem when handling multiple domains (i.e., tasks) because these tasks are domain-specific. To address this issue, Cuayahuitl et al. [18] propose a three-stage method to learn a multi-domain dialogue policy: (1) They first design a global network to connect multiple DQN networks, named *NDQN*, which can learn multiple policies for each domain. (2) They then compress raw inputs to reduce the dimension of state representations with delexicalized sentences and synonym words [29]. (3) They pre-train NDQN with collected human example dialogues to bootstrap the network, which considerably reduces the time to converge [2]. Although the three stages can be applied independently, their combination is much more efficient when scaling to multiple domains. To evaluate the effectiveness of the proposed NDQN, they build a multi-domain task-oriented dialogue system that can book restaurants and hotels at the same time. The experimental results show that the cluster of DQNs (i.e., NDQN) significantly outperforms a single DQN network in terms of task completion rate, rewards, and learning speed (up to 7 times faster than DQN). Besides, Cuayahuitl et al. [18] also compare NDQN with a K-Nearest Neighbour baseline model. It turns out that the dialogues produced by NDQN are much more successful and efficient.

Nevertheless, these reinforcement learning approaches mentioned above still have drawbacks. They straightforwardly apply existing reinforcement learning models to train dialogue policies, without addressing the characteristics of policy learning. For example, task-oriented systems usually have large action spaces and sparse rewards, which may lead the model to become trapped into local optima. In this thesis, we aim to address the problems in existing reinforcement learning models and propose corresponding solutions to tackle them.

2.2 Adversarial Dialogue Policies

In this section, we review related studies related to handling long dialogues. Particularly, we detail rule based, retrieve based, supervised learning based, and reinforcement learning based approaches in Section 2.2.1, Section 2.2.2, Section 2.2.3, and Section 2.2.4, respectively.

2.2.1 Rule Based Approaches

To handle dialogue policy learning, early studies straightforwardly build a large number of hand-crafted rules by human experts [89, 16, 55, 72, 4, 71, 84]. They first pre-define a set of templates as candidate responses. When generating responses, they usually choose the most similar response (e.g., cosine similarity) from the pre-defined templates according to given dialogue contexts and user utterances. The most two representative rule-based response models are *Alicebot* [84, 71] and *Elizabot* [89].

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity), also referred to as Alicebot, is built with a set of artificial intelligence markup language (AIML) templates [84]. It is freely available at `http://www.alicebot.org`. Alicebot can allow users to customize responses using a priority score and will output a scalar confidence score for each response. The implementation of Alicebot is simply based on *if-do* triggers. For example, if a response is a correct sentence, then it outputs a large confidence score; otherwise, it outputs a small confidence score. Algorithm 1 illustrates the procedure.

Algorithm 1 Alicebot

Input: dialogue context s and user utterance u
response $r \leftarrow$ search AIML templates using s and u
if r is a correct sentence **then**
    if r has priority p **then** confidence $c \leftarrow 1.0$
    else confidence $c \leftarrow 0.5$
else confidence $c \leftarrow 0.0$
Output: response r, priority p, and confidence c

Table 2.2: Example templates of Elizabot

Id	Template	Response
1	"I am {placeholder}"	"Did you come to me because you are ..."
2	"What {placeholder}"	"Why do you ask?"

Elizabot was firstly proposed by Weizenbaum [89], which is very similar to Alicebot. Elizabot also chooses responses from a set of pre-defined templates using string match. The difference is that Alicebot is built on an open domain while Elizabot is built to mimic a Rogerian psychotherapist and its responses are related to personal questions. Table 2.2 shows two example templates of Elizabot. Note that "{placeholder}" in the templates can be replaced with specific words. Given

a dialogue context and a user utterance, Elizabot will choose a response via string match. If there are multiple matched responses, Elizabot will randomly pick one to respond the user.

Alicebot and Elizabot, however, are both very limited in handling long dialogues because it is difficult to design such a large number of rules to cover various user utterances. Besides, neither Alicebot nor Elizabot truly understands user inputs since they did not take into account the semantic meaning. Therefore, we only use rule based approaches as one of the baselines in our thesis.

2.2.2 Retrieval Based Approaches

The intuition of retrieval based response models is similar to *learning to rank* [13, 14]. These models first compute a ranking score $s = score(context, utterance)$ for all candidate responses, and then select the response with the largest score to respond users [51]. We will discuss a popular retrieval-based response model named *Latent Variable Hierarchical Recurrent Encoder-Decoder* (VHRED) [69] in this section.

VHRED is proposed by Serban et al. [69] to generate sequential data (e.g., dialogue) with hierarchical structures in an end-to-end manner. A natural language dialogue usually contains at least two structure levels: (1) word level — this structure is mainly composed of the statistics of words within an utterance, such as word frequency and work co-occurrences [28]. (2) semantic level — this structure is characterized by the semantic meaning of the dialogue, such as dialogue topics and user intents [101]. Traditional models only consider the word level regardless of the dialogue topics and user intents, which may lead to the incoherence of user topics and some misunderstandings when parsing user intents [62]. Serban et al. [69] address this issue and propose the VHRED model to overcome it. VHRED incorporates a hierarchical latent variable RNN architecture to generate dialogue responses in both the word level and the semantic level. It is trained as a kind of sequence-to-sequence variational auto-encoder joint with a continuous high-dimensional Gaussian latent variable attached to each dialogue utterance. The training process of VHRED models is illustrated as follows: it first computes the cosine similarities between the current dialogue context and all dialogue contexts in the training dataset using average Glove [61] word embeddings or Word2Vec [56] word embeddings. It then retrieves a set of K (e.g., $K = 20$) responses according to the computed similarities as candidate responses. After that, the candidate responses are fed

into the VHRED model to compute the log-likelihood for each response. Finally, the VHRED model can return the response with the largest log-likelihood to users. Furthermore, Lowe et al. [52] extend from the VHRED model and propose a dual encoder model [3] that can utilize two separate sequence encoders ENC_Q and ENC_R to encode a dialogue context and a candidate response, respectively. They use a bi-linear mapping between a dialogue context embedding and a candidate response embedding to compute the score for each candidate response. In this way, the semantic meaning of a dialogue can be better understood.

Nevertheless, retrieval based models are very sensitive to the choice of K and thus will lead to variances when selecting responses. Besides, retrieval based models require that all responses must be listed in a candidate set before training. In the real world, dialogue responses are often associated with user inputs. If a user informs unseen entities to the dialogue system, retrieval based models may not know how to respond the user because these entities are not included in the candidate set. These drawbacks make it difficult to put retrieval based models into practice. Therefore, in this thesis, we only use retrieval based models as one of the baselines.

2.2.3 Supervised Learning Based Approaches

Supervised learning approaches have been widely used for task-oriented dialogue policy learning. Levin et al. [42] first formulate policy learning as a *Markov decision process* (MDP). However, such a formulation is very limited in handling long dialogues because the dialogue state in a certain dialogue turn cannot be fully observed. When a dialogue gets longer, the dialogue states in historical turns are hard to maintain. Then, Williams and Young [96] update policy learning to a *partially observable Markov decision process* (POMDP) which can handle long-term memories. Based on the POMDP formulation, a growing number of researchers begin to study various problems in task-oriented dialogue policy training [64, 11, 73, 67, 15]. In this section, we will detail related studies for supervised learning based approaches.

As mentioned earlier, a typical task-oriented dialogue system requires several sub-components such as NLU, policy learning, and NLG [64], while it is time-consuming and labour-intensive to train different models for each component. Traditional models are encountering two challenges in task-oriented dialogue system training: (1) the errors produced in previous dialogue turns may propagate in the whole dialogue and thus affect the decision making (i.e., action selection) of

the subsequent turns. (2) it is not an easy task to carry the knowledge captured in the previous dialogue turns into the current turn due to the long dialogue history (context). To address the first problem, Chen et al. [15], Bordes and Weston [11] propose an end-to-end task-oriented dialogue system in a different way. Rather than explicitly requiring sub-components, their models fuse NLU, policy learning, and NLG into an overall end-to-end model. Eventually, there is only one single end-to-end model that needs to be trained in their system, which significantly saves human labour and improves efficiency. However, this final end-to-end model should be able to handle user input in both word level (i.e., to understand user utterance with slot-filling) and sentence level (i.e., to select the next action), which makes it more difficult for the model to memorize the long-term dialogue history. So as to overcome this problem, *memory networks* are proposed to improve the memory of the end-to-end model [75, 15, 11]. Memory networks are a recent class of deep learning models that have been widely used in natural language processing (NLP) tasks, such as question answering [91] and language modeling [75]. Memory networks are extended from recurrent neural networks but explicitly augmented with external memory storage blocks that can be read and written to encode knowledge. Such storage blocks allow multiple computational steps (i.e., "hops") and can model long-term memory dependencies in sequential utterances. Sukhbaatar et al. [75] incorporate the *attention* mechanism [5, 82] into memory networks, which can further enhance the long-term memory of end-to-end models.

Although these supervised learning based approaches have achieved a great improvement in dialogue policy learning, they still have many limitations. For example, most of these studies use deep learning models to train dialogue policies. As we know, deep learning is data-hungry and needs a large number of training dialogues, which will cost massive human efforts in data collection. Besides, users usually would like to give ratings as the feedback after using dialogue systems but supervised learning models do not have the ability to learn the feedback information. In this thesis, we aim to address these two issues by using reinforcement learning techniques.

2.2.4 Reinforcement Learning Based Approaches

As mentioned above, there are two significant drawbacks in supervised learning models: (1) Supervised learning models are data-driven and usually data-hungry. Thus, a large number of training dialogues are needed to build a sophisticated supervised learning model. Whereas it is difficult to

obtain such dialogue data because we need two people to play the role of customers and dialogue systems. (2) Supervised learning model cannot be adaptive to user feedback. In a real scenario, after using a customer service, the user is usually requested to give a rating as feedback. With the feedback, we can continually finetune the model to adapt to user feedback, which can improve the user experience of using dialogue systems. Unfortunately, supervised learning models do not have such kind of ability. To address these two problems, reinforcement learning methods are recently proposed to reduce the number of needed training dialogues and automatically finetune the model towards the direction of obtaining better user feedback [97, 21, 19, 104, 18, 98, 96, 74]. Next, we will discuss several task-oriented dialogue systems built by reinforcement learning.

Zhao and Eskénazi [104] propose an end-to-end reinforcement learning task-oriented dialogue system with a 20-Question-Game conversational simulator. Their model is based on Deep Recurrent Q-Networks (DRQN) [26] and can jointly optimize NLU and policy learning. Besides, they overcome two problems existing in conventional dialogue systems. The first one is the *credit assignment* problem. In a real environment, dialogue users only give feedback to evaluate the quality of system at the end of the dialogue, making it difficult to analyze the errors produced in each component. This is because the errors produced in previous components will propagate to the subsequent components in the rest of the pipeline, and it is not an easy task to find out these errors. The second problem is *process interdependence*, which makes online adaptation challenging. For example, if a component such as NLU is re-trained with new data, all other components that rely on it will become sub-optimal and need to be re-trained as well. This is because the subsequent components are built on the output of the older version of the components. Therefore, they propose a new model to jointly optimize NLU and policy learning. They also introduce DRQN in their model because DRQN is very popular in dealing with POMDP problem. Moreover, to improve the learning speed, they combine the strength of supervised learning and reinforcement learning, and use supervised learning to warm-up the DRQN model.

Dhingra et al. [19] make a further improvement and build a task-oriented dialogue system that can carry out complex tasks that need to interact with users through multiple dialogue turns, such as booking movie tickets. Existing dialogue systems, such as Apple Siri and Microsoft Cortana, can only conduct simple tasks (e.g., looking up the weather or aimlessly chit-chatting with users). But these dialogue systems are very limited when the tasks become more complicated, in other

words, the length of dialogues will grow significantly. They propose a task-oriented dialogue system that can interact with users through multiple dialogue turns. Moreover, their system is able to search back-end database without explicit SQL queries. Traditionally, previous task-oriented dialogue systems need explicit SQL queries to interact with a database and to retrieve entities. However, the approach breaks the differentiability of the whole model when training in an end-to-end manner. Dhingra et al. [19] address this drawback and propose a "soft" posterior distribution over all entities in the database to replace symbolic queries. Such a "soft" posterior distribution represents the probability of all users' interest in each entity. They use reinforcement learning to learn the soft retrieval model, and thus considerably enhance the task completion rate. Furthermore, because of using reinforcement learning techniques, their model can be easily adaptive to user preferences according to their feedback.

However, these studies still have limitations. For example, they just simply apply existing reinforcement learning models to policy learning while these models have many pitfalls such as the sparse rewards problem (detailed in Chapter 5). These pitfalls will cause ceilings over reinforcement learning models and impair the task completion rates. Therefore, in this thesis, we aim to solve these pitfalls and to further improve the performance of policy learning.

2.3 User Simulator

User simulators are playing an important role in dialogue policy learning by reinforcement learning models [97, 21, 19, 104, 18, 98, 96, 74]. First, when training in the reinforcement learning fashion, a dialogue system needs to interact with an *environment*. Therefore, the traditional dialogue dataset cannot be directly applied. Second, task-oriented dialogue systems are usually domain-specific, meaning that we have to re-collect training data when building a dialogue system in a new domain. Third, it is labour-intensive and expensive to collect a large number of example dialogues for training. Fortunately, user simulator can be used to solve the aforementioned problems. In particular, at the beginning of the training process, a reinforcement learning agent tries to interact with a user simulator with a random initial policy, which produces a large number of dialogues and corresponding rewards for each dialogue to indicate the success or failure of tasks [44]. Note that in the early stage of training, most of the generated dialogues will fail

because the agent is learning from scratch. With the increase of simulation steps, the agent will tend to choose the response that can contribute to larger rewards. At the end of training, the agent can become well-trained to be deployed to a real world. Li et al. [44] propose a new simulation framework which has been released to the public. Their simulator is constructed on the domain of booking movie tickets with heuristic rules and moderate collected dialogues. Furthermore, they demonstrate their proposed user simulator with multiple dialogue agents, which proves that their framework can be easily customized. In this thesis, we follow the work of Li et al. [44] and build a new user simulator for dialogue policy learning by reinforcement learning.

2.4 Summary

In this chapter, we have reviewed several studies related to context-uncertainty-aware dialogue policies and adversarial dialogue policies. Although existing studies have achieved impressive progresses in task-oriented dialogue system training, no existing solutions can be effectively applied to handle ambiguous user inputs or long dialogues. This suggests a large room to improve, especially for reinforcement learning based approaches, which motivates us to deeply explore reinforcement learning techniques for dialogue policy learning in the following chapters.

This page is intentionally left blank.

Chapter 3
Preliminaries

In this chapter, we explain preliminary techniques related to the studied problem in detail. We first present an overview of reinforcement learning models in Section 3.1 and then discuss generative adversarial nets (GANs) in Section 3.2. Finally, we summarize this chapter in Section 3.3.

3.1 Reinforcement Learning

Reinforcement learning has been formulated as a *Markov decision process* (MDP) [88]. It can be expressed as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ denotes a finite set of state spaces; $\mathcal{A}$ denotes a finite set of action spaces; $\mathcal{P}$ denotes the transition probability $P(s_{t+1}|s_t, a_t)$ from state s_t to s_{t+1} after taking action a_t ($s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$); $\mathcal{R}$ denotes the reward function $\mathcal{R}(s_t, a_t)$ by taking action a_t on state s_t; $\gamma \in (0, 1]$ denotes the discount factor. The *return* R_t is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ to compute the cumulative discounted rewards from step t onwards, where $r_t = \mathcal{R}(s_t, a_t)$. An action-value function $Q^{\pi_\theta}(s, a) = \mathbb{E}_{\pi_\theta}[R_t|s_t = s, a_t = a]$ is defined to compute the expected return of taking action a_t on state s_t, and a state-value function $V^{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}[R_t|s_t = s]$ is defined to compute the expected value of state s_t, where π_θ is a *policy* representing how to choose the action on state s_t. The goal of reinforcement learning is to maximize the value of $Q^{\pi_\theta}(s, a)$ and to obtain an optimal policy π_{θ^*}.

There are two important concepts in reinforcement learning models: *environment* and *agent*. The environment is a kind of objects that can be formulated as MDP while the agent is a model representation of how to interact with the environment. Figure 3.1 shows a typic framework of reinforcement learning based task-oriented dialogue systems. It consists of two components: a user simulator (corresponding to the environment) and a dialogue agent. The user simulator can

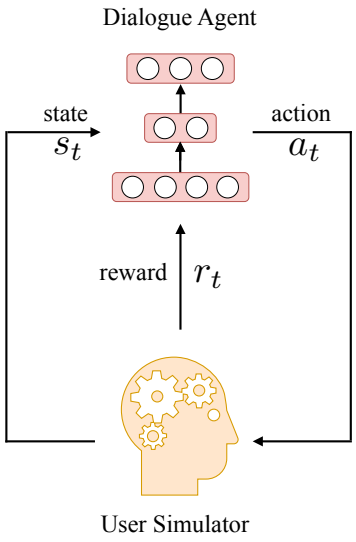


Figure 3.1: A reinforcement learning framework

simulate users' behaviours when using dialogue systems and interact with the dialogue agent, such as informing preferences, requesting information, and checking the success of user tasks [44]. At each dialogue turn t, the dialogue agent responds the user simulator with the action a_t. Meanwhile, the user simulator informs the current dialogue state s_t and provides a reward r_t to the dialogue agent. With a large number of iterations, the dialogue agent is able to get improved and finally complete users' tasks.

In particular, there mainly are three classes of reinforcement learning algorithms, including value-based, policy-based, and actor-critic algorithms. We will detail these three classes of algorithms in the following sections.

3.1.1 Value-Based RL

So as to learn the optimal policy, value-based reinforcement learning will first estimate the value of taking each possible action on a state, and then choose the action with the largest value. In this section, we will introduce several popular value-based reinforcement learning models.

Q-learning

A typical value-based algorithm is *Q-learning*, which belongs to model-free reinforcement learning. Here, Q denotes the action value, which can be defined as an action-value function $Q^{\pi}(s, a)$, where a denotes an action, s denotes a state, and π denotes a policy to be learned. The optimal action-value function can be defined as follows:

$$Q^{*}(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t{=}s, a_t{=}a, \pi] \tag{3.1}$$

Correspondingly, the optimal policy can be computed according to the obtained optimal action-value function:

$$\pi^{*}(s) = \text{argmax}_{a} Q^{*}(s, a). \tag{3.2}$$

Via combining Eq. 3.1 and Eq. 3.2, we can obtain the final formulation to search the optimal policy, which is known as *Bellman equation* [88]:

$$Q^{*}(s_t, a_t) = \mathbb{E}_{s_{t+1}} \left[r_t + \gamma \max_{a_{t+1}} Q^{*}(s_{t+1}, a_{t+1}) | s_t, a_t \right] \tag{3.3}$$

Here, s_{t+1} is the state in the next time step $t + 1$, and a_{t+1} is all available actions for s_{t+1}. Theoretically, Eq. 3.3 can be solved using dynamic programming if the cardinality of state spaces is not too large.

DQN

Deep Q-Network (DQN) is the combination of deep learning such as convolutional neural networks (CNNs) and Q-learning, which can implement a human-level control agent in an end-to-end manner. As we mentioned above, dynamic programming methods can only be used to solve Q-learning for a low-dimensional state space. Unfortunately, in a real scenario, the state spaces of most cases are often huge, and it is extremely challenging for dynamic programming models to directly learn from these high-dimensional inputs (e.g., raw image pixels). A traditional approach to handle this case is to manually design some features to reduce the dimension of inputs, but this approach will cause a ceiling on the model performance due to the bias in the feature selection stage. Recently, CNN has been widely applied in image classification by incorporating feature extraction and classification into a single deep learning model. Doing so can considerably reduce the noises in handcrafted feature selection [38, 70]. Inspired by these studies, Mnih et al.

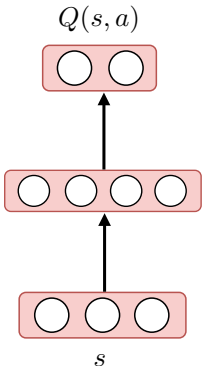


Figure 3.2: DQN model

[58] proposed DQN by integrating deep learning with Q-learning. Consequently, DQN is capable of handling the raw image data (i.e., pixels) and thus can reach the human-level control. In particular, DQN can be trained with RGB images using stochastic gradient descending methods, and can learn a state representation directly mapping raw image inputs into actions via function approximation [47, 6, 7, 63].

To train the DQN model, Mnih et al. [58] introduce *experience replay* buffer D. They store the sequences of experience tuples (s_t, a_t, r_t, s_{t+1}) into D at each time step t, where s_t is the state at step t, a_t is the action at step t, r_t is the reward at step t, s_{t+1} is the state at the next step $t + 1$. Before training the DQN model, they will first fill the reply buffer D with experience tuples via a random policy. Then, they will randomly sample a batch of experience tuples from the replay buffer to feed into the DQN model and update model parameters according to Q-learning optimization objectives. During the training process, they apply the ϵ-greedy policy to select actions. Their approach to training DQN is called *off-policy* learning (i.e., sample historical experiences to update the current policy). Compared with *on-policy* learning that uses the most recent consecutive experiences, off-policy learning is able to reduce the variances in updating model parameters because of reducing the correlations between two experiences. They evaluate the DQN model on 2600 Atari games and experimental results show that DQN significantly outperforms the best human player over 75%. Cuayáhuitl [17], Cuayahuitl et al. [18], Lipton et al. [48] propose to adapt DQN to dialogue policy learning and achieve significant improvements.

(a) Screenshot at $t1$

(b) Screenshot at $t2$

Figure 3.3: An example of POMDP environment.

DRQN

Although DQN has achieved a great performance on Atari games [58], it still has many limitations when handling those long-term historical states. Mnih et al. [58] train the DQN model using a stack of last four image frames as the input of the current step t, i.e., $s_t = [x_t, x_{t-1}, x_{t-2}, x_{t-3}]$, where x_t denotes the raw image pixels of time step t. This approach makes it challenging for DQN to memorize the longer history such as $[x_{t-4}, x_{t-5}, ...]$. More importantly, if an environment needs

to refer to the frames earlier than 4 frames, the reward will become non-Markovian. It is because, in a Markovian model, the rewards only rely on the current input. Since DQN uses the formulation of MDP, it does not have the ability to handle the underlying representation of system states. In other words, DQN [58] only works when the observations are identical to true system states.

In order to address this problem, Hausknecht and Stone [26] propose to use *partially observable Markov decision process* (POMDP) to solve the non-Markovian rewards. Compared with MDP, POMDP is able to learn the state representation of environments in a dynamic manner. It considers that the observations captured by a reinforcement learning agent are just parts of the whole environment state. Therefore, POMDP can be defined as a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O})$. Similar to the MDP formulation, here $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma$ denotes the states, actions, transition probability, rewards, and discounted factor as mentioned before. The difference is that additional Ω and $\mathcal{O}$ are introduced in POMDP, where Ω denotes the whole system state set and $\mathcal{O}$ denotes a probability distribution with respect to Ω. The agent will receive an observation $o \in \Omega$ that can be generated according to $o \sim \mathcal{O}(s)$.

In a real environment, the observations received by the agent are often incomplete and are only a part of the entire system states (i.e., POMDP). Taking driving as an example, Figure 3.3 shows two screenshots of playing a racer game, where only a part of the road appears in the driver's view at a certain time. As the car keeps going, the driver's view will become significantly different. For example, in Figure 3.3a, there are many billboards and coconut palms on the left side and right side along the roads, respectively. However, when the car gets to Figure 3.3b, the coconut palms now become a broad range of banian forest. Besides, there come some walls and stones on the right of the road. It is of importance for drivers to predict the direction of the road ahead that does not appear in the drivers' view. Hausknecht and Stone [26] find that the performance of traditional DQN model [58] will encounter a dramatic drop when the observations received by the agent are incomplete. To address this problem, Hausknecht and Stone [26] therefore introduce the POMDP formulation into DQN and propose a new model *deep recurrent Q-network* (DRQN). They systematically fuse a long short-term memory (LSTM) [33] with DQN, which allows DRQN to deal with partial observability (i.e., incomplete observations). They evaluate the DRQN model on many POMDP games and experimental results show that DRQN significantly outperforms the original DQN model. Zhao and Eskénazi [104] propose to use DRQN for dialogue policy learning

and their experimental results also show that DRQN significantly outperforms DQN, which turns out that DRQN can better capture partial observations.

Double DQN

The conventional Q-learning [88] and DQN [58] often overestimate the action values in some certain scenarios, causing that the reinforcement learning agents suffer from suboptimal policies. It is harmful to the performance of DQN models. Van Hasselt et al. [81] aim to address this problem and propose a *double deep Q-network* (Double DQN) by combining double Q-learning and DQN to solve it. Existing studies indicate that overestimation happens on account of inflexible function approximation [79] and training noises [25]. Van Hasselt et al. [81] further observe that except for these reasons, overestimation also occurs if the action values given are not reliable, no matter where the approximation errors come from. In real scenarios, overestimation is very common because no models are perfect, but it will lead to a suboptimal policy and may fail the tasks. To justify their assumption, they deeply explore the original DQN model [58] with Atari 2600 games, and find that the original DQN model usually encounters overestimations when dealing with most of these games. Therefore, they propose to introduce double Q-learning [25] that can be integrated with function approximators such as deep neural networks.

Conventional Q-learning and DQN (cf. Eq. 3.3), always use a single set of weights to select and evaluate actions, which may cause the agent to overestimate the action values. In order to address this issue, Hasselt [25] separate action selection and action evaluation by learning two value functions with two sets of weights θ and θ', respectively, where θ will be used to decide the policy to select the next action, and θ' will be used to compute the action value. Both θ and θ' can be updated with the experiences tuple in the replay buffer. Therefore, Q-learning can be re-formulated as follows:

$$y_t^{\mathrm{Q}} = R_{t+1} + \gamma Q(s_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a; \boldsymbol{\theta_t}); \boldsymbol{\theta_t}) \tag{3.4}$$

Correspondingly, Double Q-learning can also be expressed as follows:

$$y_t^{\mathrm{DoubleQ}} = R_{t+1} + \gamma Q(s_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a; \boldsymbol{\theta_t}); \boldsymbol{\theta'_t}) \tag{3.5}$$

Although Double Q-learning attempts to alleviate the problem of overestimation by separating the learning procedure into action selection and action evaluation, it still uses the same network to

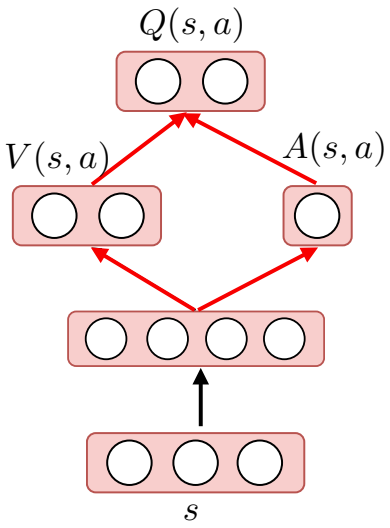


Figure 3.4: Dueling DQN

learn the weight of θ and θ' alternatively. This approach cannot truly solve the problem of overestimation. Van Hasselt et al. [81] thus propose a new network architecture named Double DQN by combining Double Q-learning and DQN. Double DQN uses an online network θ to evaluate the greedy policy and utilizes a target network θ^- to estimate the action values. Consequently, the parameters of Double DQN can be updated according to:

$$y_t^{\text{DoubleDQN}} = R_{t+1} + \gamma Q(s_{t+1}, \underset{a}{\text{argmax}}\, Q(s_{t+1}, a; \boldsymbol{\theta_t}); \boldsymbol{\theta_t^-}) \tag{3.6}$$

Compared with Double Q-learning (i.e., Eq. 3.5), the target network now becomes an independent network with the weight θ^- and will periodically synchronize from the online network θ. This approach only takes a little cost to integrate DQN with Double Q-learning but extends the effective computation from DQN. Furthermore, Van Hasselt et al. [81] compare Double DQN with vanilla DQN on playing Atari 2600 games. They find that Double DQN can not only estimate action values more accurately, but also obtain much larger scores in most of Atari 2600 games. It proves that Double DQN can reduce the overestimations in vanilla DQN and thus can learn better policies.

Dueling DQN

Even if deep learning and reinforcement learning become increasingly popular, most of these models are using traditional network structures, such as CNN, LSTM, or auto-encoders. These ordinary structures usually limit the performance of models. Wang et al. [87], therefore, propose a novel network architecture for reinforcement learning named *dueling deep Q-network* (Dueling DQN). Dueling DQN is composed of two independent estimators: (1) a state value function $V^\pi(s)$ to estimate the value of the given state s, and (2) an advantage function $A^\pi(s,a)$ to estimate the advantage after executing the action a on the state s. Note that $V^\pi(s)$ and $A^\pi(s,a)$ share a common CNN structure before being divided into two channels. Then, these two channels will be aggregated into a final action value estimator $Q^\pi(s,a)$ to estimate the value of executing the action a on the state s. The full model structure is shown in Figure 3.4. Compared to vanilla DQN proposed by Mnih et al. [58], Dueling DQN replaces the single channel $Q^\pi(s,a)$ in vanilla DQN with two aggregated channels $V^\pi(s)$ and $A^\pi(s,a)$. Such a dueling structure allows reinforcement learning agents to capture valuable states with function approximation without the need to list the combination of all available states and actions. Recall that the action value function Q is defined as:

$$Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t{=}s, a_t{=}a] \tag{3.7}$$

and the state value function is defined as:

$$V^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t{=}s] = \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)] \tag{3.8}$$

Based on these two functions, Wang et al. [87] propose an *advantage* function $A^\pi(s,a)$:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s,a) \tag{3.9}$$

The advantage function represents the difference between the action value $Q^\pi(s,a)$ and the state value $V^\pi(s,a)$, which computes the relative importance of each action on state s. Therefore, with introducing the advantage function $A(s,a)$, Dueling DQN can be expressed as follows:

$$Q^\pi(s,a;\theta,\alpha,\beta) = V^\pi(s,a;\theta,\beta) + A^\pi(s,a;\theta,\alpha) \tag{3.10}$$

where θ denotes the parameters of the CNN layers, α denotes the parameters of the advantage channel, and β denotes the parameters of the state value channel

Note that Eq. 3.10 will encounter the problem of unidentifiability that V and A cannot be reconstructed with a given Q. To overcome this problem, Wang et al. [87] force the advantage

function to produce zero advantage for the selected action by replacing Eq. 3.10 with:

$$Q^{\pi}(s,a;\theta,\alpha,\beta) = V^{\pi}(s,a;\theta,\beta) + (A^{\pi}(s,a;\theta,\alpha) - \max_{a' \in A} A^{\pi}(s,a';\theta,\alpha)) \tag{3.11}$$

or with:

$$Q^{\pi}(s,a;\theta,\alpha,\beta) = V^{\pi}(s,a;\theta,\beta) + (A^{\pi}(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a' \in \mathcal{A}} A^{\pi}(s,a';\theta,\alpha)) \tag{3.12}$$

Finally, Wang et al. [87] evaluate the Dueling DQN model with the Atari 2600 games, and the experimental results show that Dueling DQN significantly outperforms vanilla DQN on a large number of Atari 2600 games.

3.1.2 Policy-Based RL

In contrast to value-based reinforcement learning models that need two stages computation (e.g., Q-value estimation and policy selection), policy-based directly learn the optimal policy with only one function approximator. In this section, we will introduce the most popular policy-based reinforcement learning algorithm *policy gradient*.

Policy Gradient

Value-based reinforcement learning models aim to learn an action value function estimator first, and then select the action with an implicit *greedy* policy according to the learned function estimator. For example, vanilla DQN [58] selects the action with the largest estimated Q value on state s. Although these value-based models can reach a satisfactory performance in some applications, they still have many limitations. First, value-based reinforcement learning can only be used to find a stochastic policy (i.e., discrete action spaces) with outputting the probability related to the action value function. Thus it lacks the capability to deal with a deterministic policy (i.e., continuous action spaces). Second, the result of action selection will be seriously affected in spite of a small change in the value function estimator. These drawbacks make it difficult for value-based reinforcement learning models to converge, such as vanilla Q-learning and dynamic programming methods [10]. Although each learning step can be well estimated, this problem will still happen after the agent changes policies.

To solve this problem, Sutton et al. [77] propose the *policy gradient* method that directly approximates the policy with an independent function estimator, instead of learning an additional

value function estimator for policy selection. For example, a policy can be expressed with a neural network that takes a state as the input and a policy as the output. Policy gradient defines two kinds of policies: (1) a stochastic policy $a = \pi(a|s;\theta)$, and (2) a deterministic policy $a = \pi(s;\theta)$. Here, a denotes the action, s denotes the state, π denotes the policy, and θ denotes the parameters of the neural network. The objective function can be defined as:

$$L(\theta) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + ...|\pi(\cdot;\theta)\right] \tag{3.13}$$

Policy gradient can optimize the objective by gradient descending methods in an end-to-end manner and directly update the policy parameters θ towards obtaining larger rewards. For a stochastic policy, the gradient can be computed with:

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial log\pi(a|s;\theta)}{\partial \theta} Q^{\pi}(s,a)\right] \tag{3.14}$$

Similarly, for a deterministic policy, if a is continuous and Q is differentiable, the gradient can be computed with:

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial Q^{\pi}(s,a)}{\partial a}\frac{\partial a}{\partial \theta}\right] \tag{3.15}$$

3.1.3 Actor-Critic RL

In Section 3.1.1 and Section 3.1.2, we have discussed several value-based and policy-based reinforcement learning models. There are many imperfections in these models. In this section, we review the most recent studies that combine the strength of both value-based and policy-based reinforcement learning, i.e., *actor-critic* methods [59, 35].

A3C

The combination of deep learning and reinforcement learning allows models to learn much more complicated state representation, but such combination is extremely unstable for online reinforcement learning models. Hence, many studies aim to alleviate this problem and propose to use *experience replay* to stabilize these models [58, 46, 26], such as DQN, DRQN, and DDPG. In particular, the states received by the agent is usually a sequence of non-stationary data, leading to strong correlation when updating the model parameters. So as to reduce such correlation, they store the sequences of observed experiences into a replay buffer, where the agent can sample a

batch of experiences to optimize model parameters. Although this approach can reduce the correlation in somewhat level, it also forces the on-policy models to downgrade to off-policy which may cause some ceilings on the performance of reinforcement learning models. Besides, experience replay costs a large amount of memory and computation to store sufficient experience tuples. More seriously, off-policy models always use the historical data produced by the older policy to update the most up-to-date model parameters, and may lead to delay in learning from the most recent observations.

To overcome the aforementioned drawbacks, Mnih et al. [59] propose a novel deep reinforcement learning model named *asynchronous advantage actor-critic* (A3C). Rather than using experience replay, they simultaneously train multiple reinforcement learning agents in multiple worker threads and each thread runs an independent environment. The gradients in each worker will be aggregated asynchronously in a certain periodic time. Such an approach is also able to decorrelate the received observations and stabilize the training process because the agents are experiencing totally distinct states in each worker at the same time. This makes it possible to continue to improve some existing on-policy models (e.g., Sarsa) and off-policy model (e.g., Q-learning).

A3C also extends the features of actor-critic algorithms that utilize a critic network to estimate the state value $V(s_t;\phi)$ and an actor network to output the policy $\pi(a_t|s_t;\theta)$.

$$V(s_t;\theta) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t\right] \tag{3.16}$$

An advantage function $A(s_t, a_t;\phi,\theta)$ is defined in Eq 3.17 to estimate the advantage of executing a_t on s_t over the state value $V(s_t;\phi)$:

$$A(s_t, a_t;\phi,\theta) = \sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\phi) - V(s_t;\phi) \tag{3.17}$$

The gradient of actor network can be computed as follows:

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial log\pi(a_t|s_t;\theta)}{\partial \theta} A(s_t, a_t;\theta,\phi) \tag{3.18}$$

The objective of the critic network is defined as follows [59]:

$$L(\phi) = (A(s_t, a_t; \theta, \phi))^2 \tag{3.19}$$

Furthermore, such parallel mechanisms are also beneficial. Traditional deep reinforcement learning models are usually computation-wasting and GPU-consuming. Mnih et al. [59] evaluate the A3C model using a number of Atari 2600 games on a single machine with multiple CPUs. The experimental results show that the A3C model trained on CPUs significantly can outperform the state-of-the-art reinforcement learning models trained with GPU. Besides, A3C also has a good performance in handling the task of continuous motor control and 3D mazes. More importantly, A3C trained on CPUs even converges much faster than those baseline models trained on GPUs.

UNREAL

Conventional reinforcement learning models search the optimal policy by maximizing extrinsic rewards (i.e., return). However, it is often difficult to obtain such extrinsic rewards in many tasks, leading to lack of guidance for agents to learn the optimal action. Although extrinsic rewards are available, the observed states may contain several different learning targets. Regarding the task of driving, the explicit learning target is to control the car (e.g., turning and braking). In fact, there are many other implicit learning targets, such as recognizing pedestrians and traffic lights. Some unsupervised learning models such as auto-encoder [83] are used to reconstruct these implicit learning targets. Predicting the image pixels of the next states is a common approach to help models better capture the environment representation. However, straightforwardly predicting the next states in the pixel level is extremely challenging, which may cause reinforcement learning models to overly focus on the state prediction rather than the real learning target and thus degrade the final performance.

To overcome this problem, Jaderberg et al. [35] propose a new reinforcement learning model named *UNsupervised REinforcement and Auxiliary Learning* (UNREAL). Compared to predicting the pixels of the next state, UNREAL utilizes three auxiliary tasks to estimate the observed states and to introduce additional pseudo rewards into reinforcement learning models. The intuition behind UNREAL is to lead the agent to learn more valuable representation in long-term.

Specifically, UNREAL consists of four networks: (1) A vanilla A3C network [59] to interact with environments. This process produces a sequence of experiences such as (s_t, a_t, r_t, s_{t+1}) which will be stored in a small replay buffer. (2) A *pixel control* network to provide pseudo intrinsic rewards. The pixel control network divides the observed states (e.g., image) into an $n \times n$ grid and maximizes the changes of each cell with an auxiliary policy $Q^{aux}(a, i, j)$, where $Q(a, i, j)$ estimates the expected value change of cell (i, j) after executing the action a. (3) A *reward prediction* network to predict the reward of the next unseen timestep given recent states. (4) A *value function replay* network to assist the A3C agent to estimate the state value. With the four networks, UNREAL is able to learn the optimal policy and the optimal value function at the same time. It utilizes auxiliary tasks to guide the agent to comprehensively understand the learning target in both long-term (i.e., estimating return) and short-term (i.e., estimating reward). Jaderberg et al. [35] evaluate the UNREAL model with several challenging games. The experimental results show that UNREAL significantly surpasses the state-of-the-art models on Atari 2600 games and a three-dimensional Labyrinth game.

3.2 Generative Adversarial Nets

In this section, we will discuss the studies related to *generative adversarial nets* (GANs).

3.2.1 Vanilla GAN

Goodfellow et al. [22] first put forward the idea of GAN. Before GAN was proposed, deep learning is mainly applied to train discriminative models with backpropagation algorithms. In contrast, generative models are not as popular as discriminative models because there are many difficulties in training generative models. For example, it is hard to track the probabilistic computations when estimating the maximum likelihood, leading to unstable and uncontrollable outputs. To address this issue, Goodfellow et al. [22] propose to use game theory to train a generative model. GAN consists of two important models: (1) a discriminative model (i.e., discriminator D) to differentiate whether an example is from real data or from a generative model, and (2) a generative model (i.e., generator G) to generate fake examples to confuse the discriminator. This process resembles a

two-player minimax game:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\Big[\log D(x)\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[\log\left(1-D(G(z))\right)\Big] \quad (3.20)$$

where $D(x)$ denotes the discriminator that estimates the probability of the sample x coming from real data; $G(z)$ denotes the generator that generates a fake sample with the noise z; $p_z(z)$ denotes a prior noise distribution. The discriminator can be updated using gradient ascending according to:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[\log D(x_i) + \log\left(1-D(G(z_i))\right)] \quad (3.21)$$

where m is the number of training samples. Correspondingly, the generator can be updated with gradient descending according to:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}[\log\left(1-D(G(z_i))\right)] \quad (3.22)$$

Goodfellow et al. [22] train several GANs using different datasets such as MNIST[39], the Toronto Face Database (TFD) [76], and CIFAR-10 [37]. Experimental results show that the images generated by GANs are much more closed to those in the original dataset, meaning that GAN significantly outperforms traditional generative models.

3.2.2 Conditional GAN

Although vanilla GAN proposed by Goodfellow et al. [22] can surpass the most state-of-the-art generative models, it still has a big limitation that the data generated by the generator cannot be controlled. For example, we can train the generator to generate a number but cannot specify the value (e.g., "7") or style (e.g., italic or bold) of the number. This limitation seriously hinders putting GAN into practice. To address this issue, Mirza and Osindero [57] propose a new variant of GAN named *Conditional GAN* that can introduce conditions into the model training process. The conditions can be data labels, data styles, or anything to distinguish samples. They incorporate conditions y into vanilla GAN and therefore obtain a new objective function:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\Big[\log D(x|y)\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[\log\left(1-D(G(z|y))\right)\Big] \quad (3.23)$$

Correspondingly, the structure of conditional GAN is illustrated in Figure 3.5. They train a conditional GAN model on MNIST dataset using the class labels as the condition y, and evaluate the

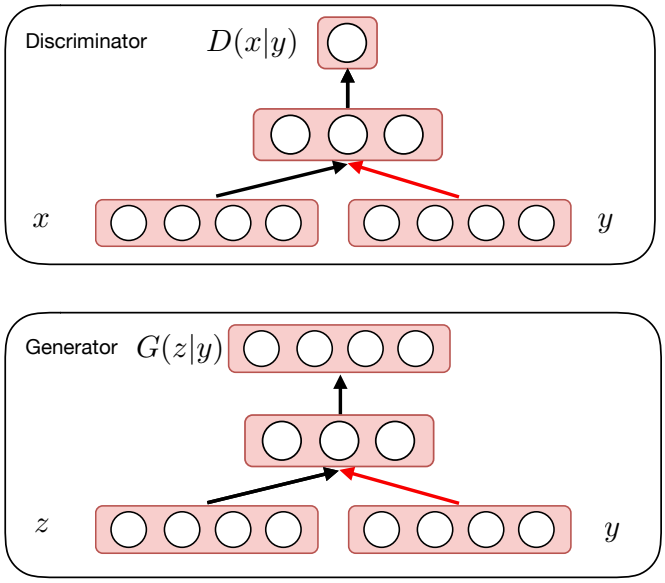


Figure 3.5: The structure of conditional GAN

model performance with the metric of Gaussian Parzen window log-likelihood. Experimental results show that conditional GAN significantly outperforms the baseline models, such as DBN [8], Stacked CAE [8], Deep GSN [9], and vanilla GAN. Furthermore, conditional GAN can also work with multimodal. For example, the photos of Flickr are also related to several user customized tags. These tags can usually describe the content of photos in a semantic level. In contrast to class labels, tags are often customized by users, which are much more diverse. They conduct preliminary experiments on MIR Flickr dataset [34] and the experimental results are satisfactory.

3.2.3 IRGAN

Conventional GAN models are often used to generate continuous data such as images. Wang et al. [85] adapt the idea of GAN to information retrieval (IR) scope and propose a new GAN paradigm named *IRGAN* to generate more general items such as documents. Similar to conventional GAN models, IRGAN also consists of two models: (1) a generative model (generator $p_\theta(d|q,r)$) and (2) a discriminative model (discriminator $f_\phi(q,d)$), where θ denotes the parameters of the generator, ϕ denotes the parameters of the discriminator, d denotes the documents, q denotes the given query, and r denotes the relevance score given by the discriminator. For instance, when handling the task of retrieving documents, the generator will predict relevant documents associated with a query by

learning a relevance distribution over all documents, while the discriminator will estimate whether the given query is relevant to the predicted documents using a pair of (q, d) as the input.

When training IRGAN, they also let the generator and the discriminator play a minimax game. In particular, the generator is trained to select relevant documents that resemble those documents in the real dataset and to obtain larger relevance scores from the discriminator. In contrast, the discriminator needs to correctly differentiate the documents selected by the generator from the ground-truth documents so as to provide more accurate relevance scores. Correspondingly, the overall objective can be given as follows:

$$J^{G^*,D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)}[\log\,(D(d|q_n))] + \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\log\,(1 - D(d|q_n))]) \tag{3.24}$$

where D can be expressed as follows:

$$D(d|q) = \sigma(f_\phi(d, q)) = \frac{\exp(f_\phi(d, q))}{1 + \exp(f_\phi(d, q))} \tag{3.25}$$

Combining with Eq 3.25, the objective of the discriminator is defined as follows:

$$\phi^* = \arg\max_{\phi} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)}[\log\,(\sigma(f_\phi(d, q_n)))] + \mathbb{E}_{d\sim p_{\theta^*}(d|q_n,r)}[\log\,(1 - \sigma(f_\phi(d, q_n)))]) \tag{3.26}$$

Note that p_{θ^*} is the current optimal parameters of the generator. It will be fixed when updating the parameters of the discriminator. Correspondingly, the objective of the generator can be defined as follows:

$$\begin{aligned}\theta^* &= \arg\min_{\theta} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)}[\log\,(\sigma(f_\phi(d, q_n)))] + \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\log\,(1 - \sigma(f_\phi(d, q_n)))]) \\ &= \arg\max_{\theta} \sum_{n=1}^{N} (\mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\log\,(1 + \exp\,(f_\phi(d, q_n)))])\end{aligned} \tag{3.27}$$

Since the document d is not continuous, the vanilla GAN formulation cannot be straightforwardly applied to compute the gradients. Therefore, Wang et al. [85] combine *policy gradient* [77] with

the generator. The derivatives of the generator gradients now become:

$$\begin{aligned}\nabla_\theta J^G(q_n) &= \nabla_\theta \, \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\log\,(1+\exp\,(f_\phi(d,q_n)))] \\ &= \sum_{i=1}^{M} \nabla_\theta p_\theta(d_i|q_n,r)\log\,(1+\exp\,(f_\phi(d_i,q_n))) \\ &= \sum_{i=1}^{M} p_\theta(d_i|q_n,r)\nabla_\theta p_\theta(d_i|q_n,r)\log\,(1+\exp\,(f_\phi(d_i,q_n))) \\ &= \nabla_\theta \, \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\nabla_\theta p_\theta(d|q_n,r)\log\,(1+\exp\,(f_\phi(d,q_n)))]\end{aligned} \tag{3.28}$$

Wang et al. [85] use the term $\log\,(1+\exp\,(f_\phi(d,q_n)))$ as a reward signal of selecting the document d for the query q_n. They evaluate the IRGAN model with three IR tasks: web search, item recommendation, and question answering. The experimental results show that IRGAN significantly outperforms the baseline models in these tasks.

3.3 Summary

In this chapter, we have discussed preliminary techniques used in our work, including reinforcement learning and adversarial generative nets. Although these techniques can be straightforwardly applied to policy learning in task-oriented oriented dialogue systems, they will encounter many problems that may cause ceilings over task completion rates, such as the problem of large action spaces and sparse rewards. In this thesis, we will explore these two problems in detail and propose efficient approaches to solve them.

Chapter 4

Context-uncertainty-aware Dialogue Policies

In this chapter, we propose a context-uncertainty-aware policy for task-oriented dialogue systems to handle ambiguous user inputs and a reinforcement learning (RL) model to learn such a dialogue policy. The proposed model is named *Parameterized Auxiliary Asynchronous Advantage Actor Critic* (PA4C). We utilize a user simulator to simulate the uncertainty of user utterances based on real data. Our PA4C model can interact with simulated users and gradually adapt to the confidence of different user utterances given a certain dialogue context. Compared with naive rule-based dialogue policies, the context-uncertainty-aware policy trained by the PA4C model can avoid handcrafted action selection and can be more robust to the variances of user utterances. The PA4C model optimizes conventional RL models with action parameterization and auxiliary tasks, which can address the problems of large action spaces and zero-reward states in dialogue policy learning We evaluate the PA4C model over training a task-oriented dialogue system for calendar event creation tasks using the proposed context-uncertainty-aware policy. Experimental results show that our model significantly outperforms the state-of-the-art RL models in terms of completion rate, dialogue length, and episode reward.

Part of the content of this chapter is published in

1. Chuandong Yin, Rui Zhang, Jianzhong Qi, Yu Sun, and Tenglun Tan. Context-Uncertainty-Aware Chatbot Action Selection via Parameterized Auxiliary Reinforcement Learning. In *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD), 2018.

4.1 Introduction

Task-oriented dialogue systems (dialogue systems for short in the rest of this chapter) [64, 19] aim to help users complete tasks of certain domains, such as making a phone call and creating a calendar event. A task may consist of several *slots*, i.e., basic elements of the task, such as *time* and *location* for a calendar event creation task. A dialogue system needs to identify these slots correctly via a dialogue with a user. Due to the uncertainty of language fluency among different people and dialogue contexts, it is not a simple task to identify these slots accurately. This problem is particularly challenging if user utterances are ambiguous, i.e., the confidence of user utterances given by *automatic speech recognition* (ASR) and *natural language understanding* (NLU) components (cf. Figure 4.1) is low. For example, the user is new to the language spoken or has a heavy accent.

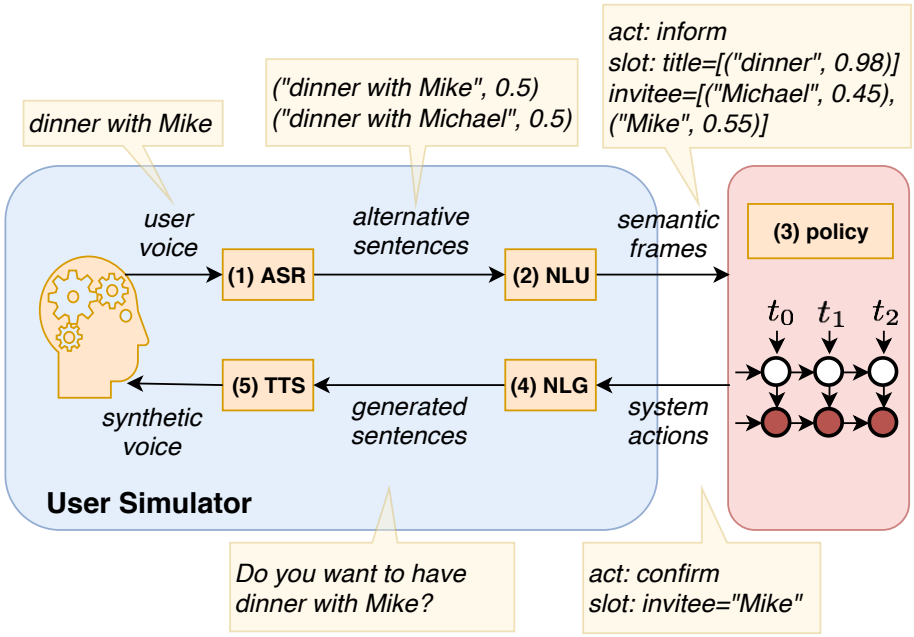


Figure 4.1: A typic task-oriented dialogue system framework.

A straightforward approach to tackle this problem is to confirm with a user whenever his or her utterance confidence is lower than a pre-defined threshold. However, in a practical scenario, this threshold may vary from person to person and is also related to a specific dialogue context. Meanwhile, users would expect dialogue systems not only to complete the tasks required, but also complete in a limited *dialogue length*. A fixed threshold cannot adapt to the diversity of

Table 4.1: Example dialogues of a rule-based dialogue system and a context-uncertainty-aware dialogue system.

Turn	**Role**	**Rule-based dialogue system**	**Context-uncertainty-aware dialogue system**
1	user	*inform* (title=[(“dinner”, 0.6)], invitee=[(“Mike”, 0.6)])	*inform* (title=[(“dinner”, 0.6)], invitee=[(“Mike”, 0.6)])
2	bot	*confirm* (title)	*confirm* (title)
3	user	*inform* (title=[(“dinner”, 0.7)])	*inform* (title=[(“dinner”, 0.7)])
4	bot	*confirm* (invitee)	*request* (time)
5	user	*inform* (invitee=[(“Mike”, 0.7)])	*inform* (time=[(“6 p.m.”, 0.6)])
6	bot	*request* (time)	*request* (location)
7	user	*inform* (time=[(“6 p.m.”, 0.6)])	*inform* (location=[(“Korean BBQ”, 0.6)])
8	bot	*confirm* (time)	*complete* ()
9	user	*inform* (time=[(“6 p.m.”, 0.7)])	
10	bot	*request* (location)	
11	user	*inform* (location=[(“Korean BBQ”, 0.6)])	
12	bot	*confirm* (location)	
12	user	*inform* (title=[(“Korean BBQ”, 0.7)])	
14	bot	*complete* ()	

user utterance confidence and may lead to lengthy dialogues, which might discourage the use of dialogue systems. To illustrate the problem, we use Table 4.1 to show two interaction sequences between a user and two dialogue systems using different dialogue policies to create a calendar event for “dinner with Mike at 6 p.m. at Korean BBQ”. The user input in these sequences is represented as an *“inform”* tuple, which contains slots including *title*, *invitee*, *time*, and *location* of the event. These slots are generated by ASR and NLU systems, which are beyond the scope of our study. Each slot is associated with a number representing the user utterance confidence proposed by the ASR and NLU systems. The “Rule-based dialogue system” column showcases how a dialogue system using a rule-based policy may interact with the user. In each turn, the rule-based dialogue system takes one of three possible actions as a response to user input: i) *confirm* (to request a confirmation of a slot previously captured from user input with low confidence), ii) *request* (to request a new slot from user), and iii) *complete* (to set up the calendar event and finish the conversation). The rule-based dialogue system confirms with the user for each slot until its

confidence reaches a fixed threshold 0.7. This has resulted in a lengthy dialogue with 14 dialogue turns.

We aim to overcome the problem of fixed confidence thresholds as illustrated above with a dialogue, which can adaptively choose a threshold according to the user utterance and the dialogue context. We call such a dialogue policy a *context-uncertainty-aware* policy. The "Context-uncertainty-aware dialogue system" column of Table 4.1 illustrates how a dialogue system using such a policy will interact with a user. This dialogue system also has a starting confidence threshold of 0.7, and it needs to confirm with the user for the first slot (*title*) that has a confidence below this threshold (Turn 2). Once this is confirmed, the dialogue system learns that only a threshold of 0.6 is sufficient to accept the input of this user. As a result, the *invitee* slot (and any slots afterwards) which also has a confidence of 0.6 does not need a confirmation anymore. This has shortened the dialogue to 8 turns and improved the user experience.

To build a task-oriented dialogue system using the context-uncertainty-aware policy, a big challenge is to collect a large amount of training data that can reflect the uncertainties of user utterances and dialogue contexts, which is labour-intensive. *Reinforcement learning* (RL) methods have been proposed to reduce the amount of data needed to train a dialogue system [97, 45]. A typical RL framework for policy learning in task-oriented dialogue system includes two key components: a *user simulator* and a *dialogue agent* to be trained. The user simulator simulates user responses to the system actions [65]. The dialogue agent can learn the best action given a certain dialogue context from such responses. The state-of-the-art results show that the task completion rates of RL-based dialogue systems have not exceeded 85% [97, 45]. This suggests room for improvement.

We take the above issues into account and propose an RL model named *PA4C* for context-uncertainty-aware dialogue policy learning. This model addresses the following two problems of existing RL models. The first problem is that, in dialogue policy learning, traditional RL models often have a large space for action selection, making it difficult to learn the best action to be selected with a large reward. The output of these models in a dialogue turn is a one-hot vector indicating which action should be selected. An action of the dialogue system consists of two components: a function (the type of actions) and its parameter (slots). For example, the action *request(time)* has a function *request* and a parameter *time*. Traditional RL models simply list

all possible combinations of functions and slots. Suppose that there are M action functions and N slots. Then the number of actions in these models will be $M \times N$. To reduce the action space and improve the reward, we introduce the *action parameterization* technique to separate the actions into two channels: one channel to learn functions and the other to learn parameters (cf. Figure 4.2b). In this way, the action space can be reduced from quadratic (i.e., $M \times N$) to linear (i.e., $M + N$).

The second problem addressed is that only a few states in dialogues have positive rewards. This makes early discovery of states that may lead to large rewards difficult. RL models thus may encounter a bottleneck due to missing valuable states. Traditional methods only focus on the target task (e.g., predict the target action) without explicitly paying attention to estimate the changes of the reward in each dialogue state. Inspired by [35], we propose to add additional tasks to the dialogue system during training and guide it to discover large-reward states (detailed in Section 4.3.2). In particular, we design two auxiliary networks: (1) a reward prediction network for predicting the reward of dialogue states, and (2) a value function replay network for helping the RL model estimate the expected state value.

This chapter makes the following contributions:

- To the best of our knowledge, we are the first to propose a context-uncertainty-aware policy for task-oriented dialogue systems that can be self-adaptive to the uncertainty of ambiguous user utterances and dialogue contexts. We also propose a novel reinforcement learning model named PA4C to train such a dialogue policy.

- To overcome the quadratic action space problem in policy learning via reinforcement learning, we propose the action parameterization technique which learns the functions and slots in two separate channels.

- We further propose two auxiliary networks to guide our model to pay extra attention to valuable states, which is more robust to both short-term immediate rewards and long-term expected returns.

- We perform experiments to verify the effectiveness of the proposed techniques. The results show that our model outperforms the state-of-the-art RL models in terms of success rate, dialogue length, and episode reward.

4.2 Preliminaries

We start with basic concepts in *deep reinforcement learning* (DRL). A DRL model is essentially a *Markov decision process* (MDP). It can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ denotes the environment state; $\mathcal{A}$ denotes the action space; $\mathcal{P}$ denotes the transition probability $P(s_{t+1}|s_t, a_t)$; $\mathcal{R}$ denotes the expected immediate reward function $\mathcal{R}(s_t, a_t)$; γ denotes a discount factor, $\gamma \in (0, 1]$ [104]. The goal of a DRL model is to maximize the *return* (cumulative expected rewards from the state s_t) $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $r_t = \mathcal{R}(s_t, a_t)$. To maximize the return, there are two approaches in general: value-based DRL and policy-based DRL.

4.2.1 Value-based DRL

Value-based DRL estimates the value of executing different actions in a state, and selects the action with the largest value. A typical value-based model is *Q-learning*, where "Q" represents the action value. Q-learning defines an action-value function $Q^{\pi}(s, a)$, where a is an action, s is a state, and π is a policy to be learned. It aims to find an optimal policy π with the maximum Q value according to $Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$ and $\pi^*(s) = \text{argmax}_a Q^*(s, a)$ [58]. *Bellman equation* [88] is used to search for the optimal policy:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}} \left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t \right] \tag{4.1}$$

Here, s_{t+1} is the next state and a_{t+1} is any possible actions for s_{t+1}.

4.2.2 Policy-based DRL

Instead of directly optimizing the policy, value-based DRL models estimate a Q value for each action and chooses the action with the maximum Q value. This approach may cause some biases. For example, assume that there are two actions a_1 and a_2 with Q values 50 and 49, respectively. The action a_2 will not be selected, although it may have a larger long-term value than that of a_1.

Policy-based DRL models are proposed to tackle this problem by directly optimizing the policy [77]. A typical approach is called *policy gradient*, which can define a stochastic policy $a = \pi(a|s; \theta)$, where θ is the weight. The total reward can be computed as:

$$\mathcal{L}(\theta) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + ... | \pi(\cdot; \theta) \right] \tag{4.2}$$

According to Sutton et al. [77], the gradients can be updated as follows:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial log \pi(a|s;\theta)}{\partial \theta} Q^{\pi}(s,a)\right] \tag{4.3}$$

4.3 Proposed Model

Most existing studies straightforwardly use traditional RL models such as DQN, DRQN, and policy gradient to train dialogue policies [97, 45]. However, these models will usually encounter a ceiling in terms of the task completion rates due to the large action space problem and zero-reward states problem in dialogue policy training. In this chapter, we aim to address these problems and propose a new RL model named *parameterized auxiliary asynchronous advantage actor-critic* (PA4C) for context-uncertainty-aware dialogue policy training. Overall, our PA4C model consists of two parts: (1) a *parameterized A3C* (PA3C) network which solves the huge action space problem in traditional RL models for policy learning; (2) two *auxiliary networks* (reward prediction and value function replay) which help PA3C discover the states with large rewards and enhance the model robustness.

4.3.1 Parameterized A3C (PA3C)

Our PA3C model is based on the vanilla A3C model proposed by Mnih et al. [59]. Recall that vanilla A3C is a hybrid RL model which combines the strength of both value-based and policy-based RL models. As Figure 4.2a shows, vanilla A3C is composed of two separate networks with shared weights: (1) an *actor* network that outputs the dialogue policy $\pi(a_t|s_t;\theta)$; (2) a *critic* network that estimates the state-value in a certain dialogue turn $V(s_t;\phi)$ [59], where θ and ϕ denotes the weights of the actor and the critic network, respectively. Note that rather than directly estimating the action-value $Q(s_t, a_t)$ like what DQN does, vanilla A3C computes a *state-value* function $V(s_t;\phi)$ and an *advantage* function $A(s_t, a_t;\theta,\phi)$ separately. Here, the state-value function $V(s_t;\phi)$ is defined to estimate the value in a certain dialogue state:

$$V(s_t;\phi) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t\right] \tag{4.4}$$

The advantage function $A(s_t, a_t; \theta, \phi)$ is defined to estimate the advantage of executing a_t on s_t over the state value $V(s_t; \phi)$:

$$\begin{aligned} A(s_t, a_t; \theta, \phi) &= Q(s_t, a_t; \phi) - V(s_t; \phi) \\ &= \sum_{i=0}^{T-1} \gamma^i r_{t+i} + \gamma^T V(s_{t+k}; \phi) - V(s_t; \phi) \end{aligned} \tag{4.5}$$

where T is the number of dialogue turns. Intuitively, the state-value function $V(s_t; \phi)$ estimates how good it is to be in a particular dialogue state s_t. In contrast, the action-value function $Q(s_t, a_t; \phi)$ estimates the value of selecting a particular action a_t in the dialogue state s_t. The advantage function $A(s_t, a_t; \theta, \phi)$ subtracts $V(s_t; \phi)$ from $Q(s_t, a_t; \phi)$ to obtain the relative importance for each available action.

Therefore, the critic network can be optimized by minimizing the mean-squared-error loss [59] as follows:

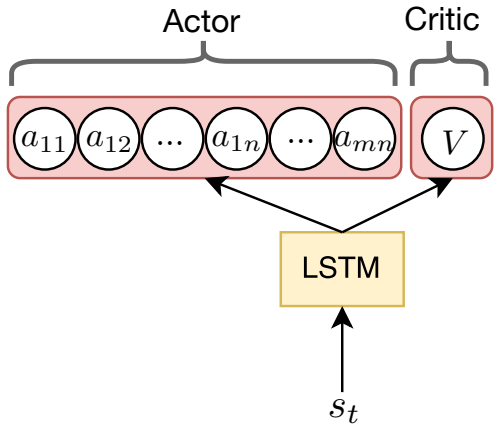
$$\mathcal{L}(\phi) = (R_t - V(s_t; \phi))^2 \tag{4.6}$$

So as to encourage the vanilla A3C model to look for more possible actions, we add an entropy regularization term $H(\pi(s_t; \theta))$ to the learning objective of the actor network [59]. Therefore, combining Eq. 4.3 and Eq. 4.5, the actor network can optimized by directly maximizing the policy as follows:

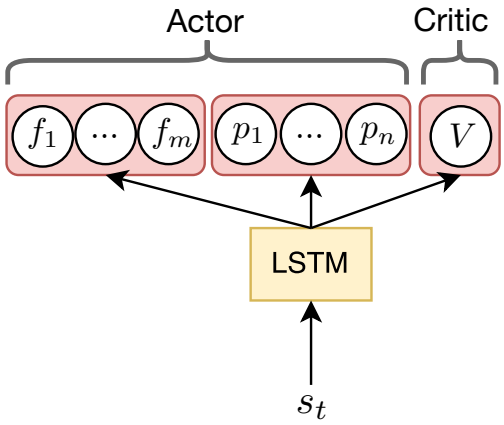
$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{\partial log \pi(a_t|s_t; \theta)}{\partial \theta} A(s_t, a_t; \theta, \phi) + \beta \frac{\partial H(\pi(a_t|s_t; \theta))}{\partial \theta} \tag{4.7}$$

However, like other RL models, vanilla A3C will also encounter the large action space problem. This is because vanilla A3C is still using a traditional model structure whose output is a one-hot vector representing the selected action. For example, assume that there are a set of M *functions* $\{f_1, f_2, ..., f_M\}$ and N *parameters* $\{p_1, p_2, ..., p_N\}$ available (functions and parameters correspond to dialogue acts and slots respectively in the user simulator). As Figure 4.2a shows, the size of final action space in vanilla A3C now becomes $M \times N$ because the actor network simply lists all combinations of functions and parameters. This will dramatically enlarge the action space to explore, which will increase the training time and may even lead RL models into the local optima.

To address this problem, we propose PA3C by introducing the *action parameterization* technique into the vanilla A3C model to overcome the large action space problem. Rather than listing



(a) Vanilla A3C model



(b) Parameterized A3C model

Figure 4.2: Model structure comparison.

all combinations of dialogue acts and slots, our model will separate the policy π in the actor network into two sub-policies π_f and π_p, which directly learns functions and parameters, respectively. As Figure 4.2b shows, the actor network in PA3C now consists of two channels (i.e., a function channel and a parameter channel) with sharing LSTM weights. In this manner, the number of actions in our model can be reduced from $M \times N$ (quadratic) to $M + N$ (linear). Furthermore, learning dialogue acts and slots in separate channels makes PA3C more flexible. For example, when slots are added, we just need to finetune the parameter channel by fixing the weights of the parameter channel, which can significantly reduce the number of neural units and thus speed up the training process.

Therefore, we re-define the optimization objective of the actor network in PA3C by replacing θ

with θ_f and θ_p that denote the weights of π_f and π_p respectively. The loss of the function channel can be optimized by gradient ascending to maximize:

$$\frac{\partial \mathcal{L}(\theta_f)}{\partial \theta_f} = \frac{\partial log \pi_f(f_t|s_t;\theta_f)}{\partial \theta_f} A(s_t, f_t; \theta_f, \phi) + \beta \frac{\partial H(\pi_f(f_t|s_t;\theta_f))}{\partial \theta_f} \tag{4.8}$$

Similarly, the loss of the parameter channel can also be optimized by gradient ascending to maximize:

$$\frac{\partial \mathcal{L}(\theta_p)}{\partial \theta_p} = \frac{\partial log \pi_p(p_t|s_t;\theta_p)}{\partial \theta_p} A(s_t, p_t; \theta_p, \phi) + \beta \frac{\partial H(\pi_p(p_t|s_t;\theta_p))}{\partial \theta_p} \tag{4.9}$$

Therefore, the loss function of PA3C can be given by summing Eq. 4.8, Eq. 4.9, and Eq. 4.6. Note that $\mathbf{\Lambda}$ is a mask vector indicating whether the function f_m has a parameter. This is because some dialogue acts do not have parameters. For example, the parameter of the dialogue act *welcome()* is void so that we need $\mathbf{\Lambda}$ as a multiplier.

$$\mathcal{L}_{PA3C} = \mathcal{L}(\theta_f) + \mathbf{\Lambda}\mathcal{L}(\theta_p) + \mathcal{L}(\phi) \tag{4.10}$$

4.3.2 Auxiliary Tasks

A recent study proposed by Jaderberg et al. [35] suggests that incorporating reasonable auxiliary tasks can improve the model robustness and performance. In the real world, most environments include more than one learning target. For example, when we train a model to play Flappy Bird in an end-to-end manner (i.e., directly mapping images to actions), there are many other underlying learning targets contained in the images, such as whether pipes appear in the current frame. However, most studies only treat the direction control of the bird as the main learning target, without consideration of other underlying targets [58]. In fact, these underlying targets are usually beneficial for RL models to learn the main target [35]. Therefore, we adapt this idea into dialogue policy learning and design two auxiliary tasks to improve the model performance.

Reward Prediction (RP)

As we know, the final goal of a dialogue agent is to produce the response that can obtain maximum global rewards. The dialogue agent needs to identify these states that can lead to large rewards and values. However, in task-oriented dialogue systems, the rewards are usually sparse. Only a

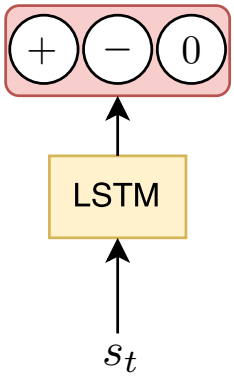


Figure 4.3: Reward Prediction (RP)

few dialogue states can provide meaningful immediate rewards. For example, large negative or positive rewards only occur when the agent finishes the task. The rewards of most intermediate states during a dialogue are zeros or very small numbers, making it difficult for RL models to learn the values of these states with large rewards. This may lower the task completion rates and cause lengthy dialogues, which will influence the user experience.

So as to overcome this problem, we decide to introduce the *reward prediction* (RP) auxiliary task to assist the learning of dialogue policies. In particular, we design an RP network $\sigma(r_{t+1}|s_t)$ to predict the immediate reward of the next unseen dialogue state given a historical context. In contrast to the state-value function $V(s_t;\phi)$ which is used to estimate returns in policy learning, the RP network is only used to help the agent pay more attention to these states with larger rewards. This makes the agent more sensitive to the changes of rewards without introducing any biases into the state-value function.

To train the RP network, we first need to save the result of the interaction between the PA3C agent and the user simulator into a replay buffer. This process will produce a number of sequences such as $(s_1, s_2, s_3..., s_T)$. Then, we sample a historical sequence $S_t = (s_{t-k}, s_{t-k+1}, ..., s_{t-1})$ from the replay buffer and predict the reward r_t of the state s_t using the sample S_t. Note that we just expect the agent to pay attention to the trend of the change of rewards rather than the specific reward values. For this sake, we decide to train the RP network to predict the sign of the rewards in three classes: *positive*, *negative* and *zero* (cf. Figure 4.3). Therefore, the RP network can be trained to minimize the cross-entropy loss as defined in Eq. 4.11, where N denotes the number of classes ($N = 3$ in the RP network), y denotes the sign of rewards in the replay buffer, and

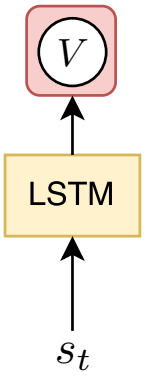


Figure 4.4: Value-function Replay (VR)

$\hat{y} = \sigma(s)$ denotes the output of the RP network.

$$\mathcal{L}_{RP} = \sum_{i}^{N} \hat{y}_i \log y_i \tag{4.11}$$

Value function Replay (VR)

As we know, vanilla A3C is trained on-policy (i.e., update the policy with the most recent states and use the updated policy to make decisions). Although vanilla A3C parallelly runs multiple agents in independent worker threads, it still cannot totally reduce the correlations in experience tuples. This is because the number of threads is usually set to less than 32 due to the limitation of CPUs. Therefore, vanilla A3C may encounter an unstable training process and become trapped into local optima, especially when training the critic network.

To enhance the stability and efficiency of the training process, we propose the *value function replay* (VR) network $V^-(s_t;\phi)$ to aid the critic network $V(s_t;\phi)$ to estimate state values (cf. Figure 4.4). We adapt the idea of the experience reply used in DQN to our work. In particular, we create a small reply buffer to store the most recent 2,000 dialogues and sample a batch of historical dialogues to update the parameters of the critic network in PA3C. Note that $V^-(s_t;\phi)$ and $V(s_t;\phi)$ share the same weights ϕ. The only difference is that $V^-(s_t;\phi)$ is trained off-policy (i.e., using the historical experiences) but $V(s_t;\phi)$ is trained on-policy (i.e., using the most recent experiences). In this way, the final state-value function $V(s_t;\phi)$ can combine the strength of both on-policy and off-policy training, making the critic network much more robust when estimating

the expected return. Similar to the training of the critic network, the VR network can also be optimized to reduce the MSE loss as follows:

$$\mathcal{L}_{VR} = \mathcal{L}(\phi) = (R_t - V^{-}(s_t; \phi))^2 \tag{4.12}$$

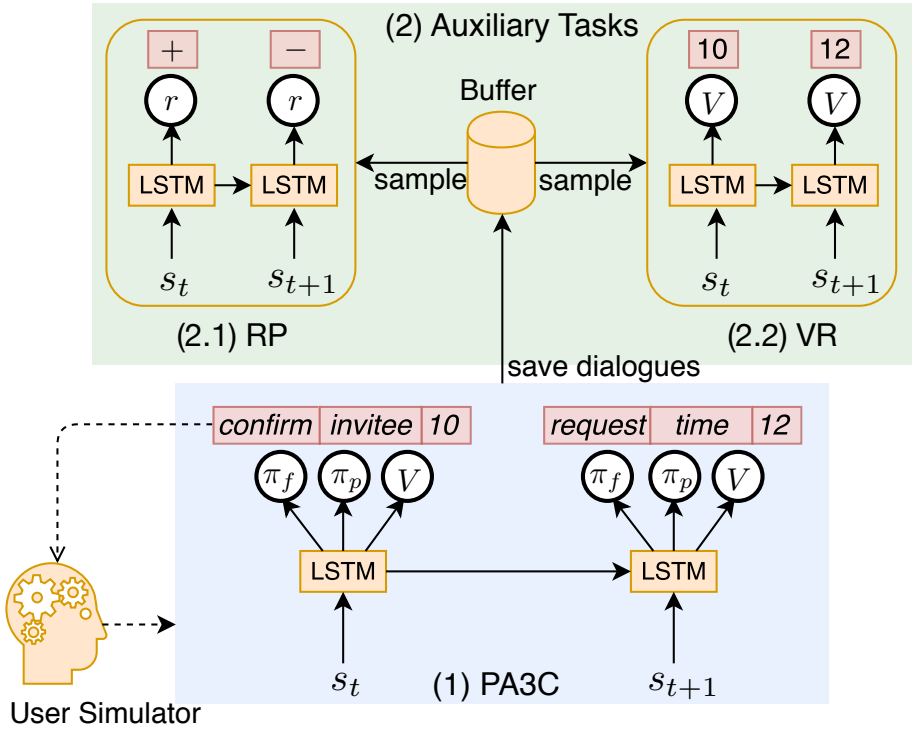


Figure 4.5: Full PA4C model consists of (1) PA3C and (2) Auxiliary Tasks (RP and VR networks). All LSTM layers share the same weights.

4.3.3 PA4C model

Our full PA4C model integrates the PA3C agent and two auxiliary networks. The full model is illustrated in Figure 4.5. Firstly, the PA3C agent interacts with the user simulator, which generates a series of dialogues. These dialogues will be stored into a small replay buffer, where RP and VR can sample historical sequences to update their weights. The final loss of PA4C is the combination of PA3C, RP, and VR networks [35]:

$$\mathcal{L}_{PA4C} = \mathcal{L}_{PA3C} + \lambda_{RP}\mathcal{L}_{RP} + \lambda_{VR}\mathcal{L}_{VR} \tag{4.13}$$

where λ_{RP} and λ_{VR} are the weight factors of the RP and VR networks respectively; $\mathcal{L}_{PA3C}$

denotes the loss of the PA3C network; $\mathcal{L}_{RP}$ denotes the cross entropy loss of the RP network; $\mathcal{L}_{VR}$ denotes the MSE loss of the VR network.

Our PA4C model also extends the asynchronous training from vanilla A3C via multiple threads. First, PA3C will create a global network in the main thread and multiple local networks in several independent threads. The global network is responsible for dispatching gradients to each local network, which will run a user simulator to compute the local gradients. At the end of each simulation, the gradients of local networks will be aggregated back to the global network. The training procedure is summarized in Algorithm 2.

Algorithm 2 Training of PA4C

Initialize a global PA4C network with the weights $(\theta_f, \theta_p, \phi)$
Initialize k local PA4C networks $[(\theta_f^1, \theta_p^1, \phi^1), ..., (\theta_f^k, \theta_p^k, \phi^k)]$
Initialize a replay buffer $D = []$
for each local network $(\theta_f^k, \theta_p^k, \phi^k)$ **do**
    Synchronize the weights $(\theta_f, \theta_p, \phi)$ to $(\theta_f^k, \theta_p^k, \phi^k)$
    Initialize a user simulator and get an initial state s
    Initialize an experience list $e = []$
    while s is not the last turn **do**
        The actor network outputs a dialogue act f with $\pi(f|s; \theta_f^k)$ and a slot p with $\pi(p|s; \theta_p^k)$
        Respond the simulator with f and p
        The simulator transfers to the new state s' and gives a reward r
        The critic estimates a state value v with $V(s; \phi^k)$
        Save (s, f, p, r, v) into e
        $s \leftarrow s'$
    Save e into D
    Optimize the PA3C agent with $\mathcal{L}_{A3C}$ using e
    Sample a sequence $S = [(s_1, r_1, v_1), ..., (s_n, r_n, v_n)]$ from D
    Optimize the RP network with $\mathcal{L}_{RP}$ using S
    Optimize the VR network with $\mathcal{L}_{VR}$ using S
    Backup the weights $(\theta_f^k, \theta_p^k, \phi^k)$ to $(\theta_f, \theta_p, \phi)$

4.4 Experiments

In this section, we compare the performance of dialogue systems trained with our PA4C model and with baseline models in six metrics: average *completion rate* (CR), average *dialogue length* (DL), average *episode reward* (ER) (the cumulative rewards of each turn in a dialogue), and their standard deviations ("std" for short in the rest of the chapter) over 10 runs, i.e., std CR, std DL, and

std ER [65, 62]. All the DRL models are trained on 180 dialogues with noises and evaluated on 120 dialogues. The result shows that our model outperforms the state-of-the-art models in these metrics for training a dialogue system in the calendar event creation task.

4.4.1 Data Preparation

To simulate the real scenarios, we develop a data collector website, where volunteers can describe their calendar events in various expressions by voice. Then, they are requested to label slots (e.g., *location*) as the ground truth. We use Google Speech API as our ASR unit and train a simple entity extraction model using bi-directional LSTM with CRF [53] based on the CoNLL-2003 dataset [80] as our NLU unit. In all, we obtain 300 calendar events from volunteers, including 300 titles, 300 time, 113 invitees and 173 locations. The max, min, and average dialogue turns are 15, 4, and 6.54, respectively. Since the uncertainty produced by ASR and NLU may cause a user's real intent to be distorted, such as "Mike" being misunderstood into "Michael", we construct a database *IntentDB* to store these intents with probabilities as shown in Table 4.3.

In addition, we calculate the mean μ and the standard deviation σ of the ASR & NLU uncertainty for each dialogue in The Dialogue State Tracking Challenge 2 (DSTC2) dataset (in the domain of restaurant booking) [93]. We save the pairs of μ and σ into a *NoiseDB*. At each step of a dialogue simulation, a Gaussian noise with μ and σ will be introduced to augment the collected data (detailed in Section 4.4.2). In this way, we can simulate millions of dialogues by adding different noises to the 300 events collected from volunteers.

Table 4.2: Available actions and slots in the user simulator.

User Actions	*inform(slot), confirm_deny(slot), confirm_accept(slot), complete(), abort(), dont_care(slot)*
System Actions	*greeting(), request(slot), confirm(slot), complete(), abort()*
Slots	title, time, invitee, location

4.4.2 User Simulator

We construct a user simulator for task-oriented dialogue system training. Our user simulator is based on the work of Li et al. [44]. As an example application, we focus on training a dialogue

Table 4.3: User intent database: the number in each tuple represents a confidence level, which is the product of ASR and NLU confidence.

id	title	time	location	invitee
1	("dinner", 0.9)	("7 p.m.", 0.95)	("Korean BBQ", 0.87)	("Michael", 0.54), ("Mike", 0.46)
...	...	...	...	...

Algorithm 3 User simulation

Initialize:
  $T \leftarrow 0$, $complete \leftarrow False$, $terminal \leftarrow False$
  $\psi \leftarrow RandomSelect(IntentDB)$ // user intent (a distribution)
  $\omega \leftarrow Sample(\psi)$ // user real goal
  $\omega' \leftarrow \{\}$ // system recorded goal
  $z \leftarrow RandomSelect(NoiseDB)$ // noise, a pair of (μ, σ)
  $\mathcal{A}_{user} \leftarrow null$ // user action
  $\mathcal{A}_{bot} \leftarrow Action.greeting$ // system action
repeat
  $\mathcal{A}_{user} \leftarrow UserRespond(\mathcal{A}_{bot}, \psi, z)$ // randomly selected with $\mathcal{A}_{bot}, \psi, z$
  $\mathcal{A}_{bot} \leftarrow ChatbotRespond(\mathcal{A}_{user})$ // outputted by RL model
  $T \leftarrow T + 2$
  $\omega' \leftarrow \omega' + ParseEntity(\mathcal{A}_{bot})$
  if $T > T_{max}$ or $\mathcal{A}_{bot} == Action.complete$ **then**
    $terminal \leftarrow True$
  $reward \leftarrow R(terminal, complete, T)$
until $terminal$
if $\omega == \omega'$ **then** $complete \leftarrow True$
$reward \leftarrow R(terminal, complete, T)$

system for calendar event creation tasks, although the techniques proposed may be applied to train a dialogue system for other tasks. All available user actions, system actions, and slots are defined in Table 4.2.

The simulation is based on the fact that a user's intent is known by himself (i.e., user simulator) but unknown by the dialogue system. A step of a simulation includes two parts: (1) the system asks the user a question, and (2) the user answers the question honestly. Each step will be assigned with a scalar reward according to the reward function $\mathcal{R}(terminal, complete, T)$ defined in the Eq. 4.14. The goal of a system is to learn to ask a user questions which can maximize the total reward of a dialogue. The intuition of $\mathcal{R}$ is to give a higher reward to states that lead to a short

dialogue that completes a user task successfully.

$$\mathcal{R}(terminal, complete, T) = \begin{cases} -0.1, & \text{if not } terminal \\ 2.0 \times \dfrac{T_{max} - T}{T_{max}}, & \text{if } terminal \text{ and } complete \\ -1.0, & \text{if } terminal \text{ and not } complete \end{cases} \tag{4.14}$$

Here, $terminal$ indicates whether the simulation finishes, $complete$ indicates whether the system completes the task, T represents the length of the current dialogue, and T_{max} represents the pre-defined maximum length of a dialogue. A simulation terminates once T_{max} steps are executed regardless whether the user task has been completed. The reward function is Markovian because it only depends on the current dialogue state (i.e. $terminal$, $complete$, and T). At the end of each simulation, we compare the user's real goal ω (i.e., the *ground truth*) with the entities ω' captured by the system. If ω is equal to ω', it means that the system completes the user's task. Otherwise, the task fails. Algorithm 3 shows the detail of a simulation.

4.4.3 Baseline Models

We implement four baseline models, including a rule-based model and three existing DRL models, i.e., DQN [45], DRQN [104] and A3C [59]. To verify the effectiveness of action parameterization and auxiliary tasks, we also design two sub-models: A4C (only adding auxiliary networks to A3C) and PA3C (only adding action parameterization to A3C).

- **Rule-based** We implement a rule-based dialogue system by *if-else* triggers. If a slot is informed by a user and its confidence is larger than a pre-defined threshold 0.724 (the average confidence of the collected events), the system will request the next slot. Otherwise, the system will confirm it with the user. When all slots are obtained, the system will finish the simulation process.

- **DQN** We stack two consecutive states as the input s_t at step t, and use two fully-connected layers with 256 and 64 hidden units, respectively. The replay buffer size is 10^5 and batch size is 128. In the first 10^6 step, we use the *epsilon-greedy* policy to explore actions, with ε annealing from 1.0 to 0.1.

- **DRQN** We only feed one single state into DRQN at step t. The replay buffer stores the full episode of a dialogue instead of the independent states in DQN. The $timestep$ of LSTM is set to 10. The remaining hyperparameters are the same as those in DQN.

- **A3C** We introduce LSTM in A3C. Unlike the vanilla A3C that uses a fixed step to update the gradients in local networks [59], we synchronize the gradients in each local network to the global network per episode. We also set the regularization of policy entropy $\beta = 0.1$ to encourage action exploration and train with 16 threads.

- **A4C** We add *auxiliary networks* into the vanilla A3C model and then can obtain the A4C model. Both λ_{RP} and λ_{VR} in Eq. 4.13 are set to 1.0. The replay buffer for auxiliary networks is 2000. When training the RP network, we clip the rewards whose absolute value is smaller than 0.1 to 0 to get their sign (i.e., positive, negative, zero). We assume these small rewards cannot guide to discover valuable states. Other settings are identical to those in the A3C model.

- **PA3C** We add *action parameterization* techniques into the vanilla A3C model. Note that the number of actions in PA3C will be different from that in vanilla A3C because of the introducing of action parameterization. Other settings are identical to those in the vanilla A3C model.

- **PA4C** The PA4C model is the integrated version of PA3C and A4C sub-models. The hyper-parameters of these two sub-models are the same as those in the A4C model and the PA3C model.

4.4.4 Hyperparameters

In the training phase, we use the same set of hyperparameter values in these models. The input feature is composed of a 52-dimensional vector provided by the user simulator, including the last system action, the current user action, and the current dialogue length. All LSTM layers have 256 hidden units, followed by an ReLU activation function. The discounted factor of reward γ is 0.99 for all models. We use the RMSProp optimizer for gradients computing with learning rate $\eta = 0.001$ and weight decay $\alpha = 0.99$. The maximum dialogue length T_{max} is set to 20. The

action *greeting()* is removed from the set of actions because *greeting()* is always called firstly by the system. The number of actions for non-parameterized models and parameterized models are $M_p \times N + M_{np} = 2 \times 4 + 2 = 10$ and $M + N = 4 + 4 = 8$, respectively, where M_p denotes the number of functions with parameters, i.e., $\{$*request(slot), confirm(slot)*$\}$; M_{np} denotes the number of functions without parameters, i.e., $\{$*complete()*, *abort()*$\}$; N denotes the number of slots, i.e., $\{$*title, time, location, invitee*$\}$; $M = M_p + M_{np}$. Note that all the reported hyperparameters are optimal for each model via the grid search technique.

Table 4.4: Performance comparison on average. Larger *completion rate (CR)* and *episode reward (ER)* indicate better performance. Smaller *dialogue length (DL)* indicate better performance. The numbers in the parenthesis shows the absolute and relative improvements over the rule-based model. When computing the improvement, we scale the value of *ER* to $[0, +\infty]$ because there are negative rewards. The results are averaged over 10 runs.

Model	CR	DL	ER
RULE	0.634	12.925	-0.286
DQN	$0.794(+0.160, +25\%)$	$\mathbf{7.075}(-5.850, -45\%)$	$0.443(+0.729, +29\%)$
DRQN	$0.691(+0.057, +09\%)$	$10.925(-2.000, -15\%)$	$-0.194(+0.092, +04\%)$
A3C	$0.843(+0.209, +33\%)$	$9.889(-3.036, -23\%)$	$0.302(+0.588, +23\%)$
A4C	$0.854(+0.211, +35\%)$	$10.402(-2.523, -20\%)$	$0.261(+0.547, +22\%)$
PA3C	$0.900(+0.257, +42\%)$	$8.382(-4.543, -35\%)$	$0.519(+0.805, +32\%)$
PA4C	$\mathbf{0.932}(+0.289, +47\%)$	$7.558(-5.367, -42\%)$	$\mathbf{0.585}(+0.871, +35\%)$

Table 4.5: Performance comparison on standard deviation. Smaller *dialogue length (DL)*, *std CR*, *std DL* and *std ER* indicate better performance. The numbers in the parenthesis shows the absolute and relative improvements over the rule-based model. The results are averaged over 10 runs.

Model	std CR	std DL	std ER
RULE	0.052	5.769	1.018
DQN	$0.041(-0.011, -21\%)$	$\mathbf{1.889}(-3.880, -67\%)$	$0.651(-0.367, -36\%)$
DRQN	$0.046(-0.006, -12\%)$	$2.429(-3.340, -58\%)$	$0.999(-0.019, -02\%)$
A3C	$0.037(-0.015, -29\%)$	$3.075(-2.694, -47\%)$	$0.560(-0.458, -45\%)$
A4C	$0.034(-0.018, -35\%)$	$3.675(-2.094, -36\%)$	$0.669(-0.349, -34\%)$
PA3C	$0.029(-0.023, -44\%)$	$2.894(-2.875, -50\%)$	$0.492(-0.526, -52\%)$
PA4C	$\mathbf{0.025}(-0.027, -52\%)$	$2.216(-3.553, -62\%)$	$\mathbf{0.469}(-0.549, -54\%)$

4.4.5 Results

Comparison with the Rule-based Model

Table 4.4 shows the performance of the dialogue systems built with the rule-based model and the proposed PA4C model, respectively. The results illustrate that our PA4C model significantly outperforms the rule-based model in all metrics, achieving 0.932 in completion rates, 7.558 in

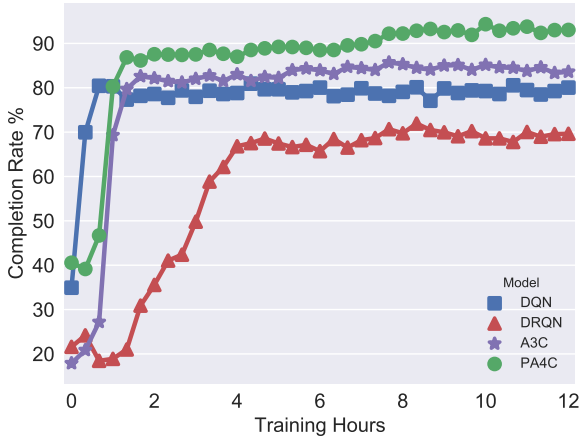
dialogue length, and 0.585 in episode rewards. Compared with the rule-based model, PA4C makes a considerable improvement in terms of all metrics, reaching 47% in completion rate, 52% in std completion rate, 42% in dialogue length, 62% in std dialogue length, 35% in episode reward and 54% in std episode reward (cf. Table 4.4 and Table 4.5). In particular, the PA4C model can achieve over 0.93 in completion rate with less than 7.6 dialogue turns to complete the task. In contrast, the rule-based model only reaches 0.63 in completion rate but takes more than 12 turns. This is because the rule-based model only uses a static threshold to confirm with users when handling ambiguous user inputs, without the consideration of the uncertainty of the current dialogue context. In contrast, the dialogue system trained with the PA4C model incorporates these uncertainties in the training phase via the interaction of a user simulator, and therefore can adapt to the uncertainty of user utterances according to the dialogue context. Such a dialogue system is called *context-uncertainty-aware* dialogue system. It can produce a high completion rate as well as a short dialogue, which can significantly improve the user experience. Table 4.5 also shows that the standard deviations in the PA4C model are much lower than the rule-based model, meaning that the dialogue system trained with the PA4C model will be more stable and more robust.

Comparison with RL Models

We compare the proposed PA4C model with the state-of-the-art reinforcement learning models such as DQN, DRQN, and A3C.

- **PA4C v.s. DQN.** From Table 4.4, we can see that PA4C outperforms DQN with improving 22% in completion rate and 6% in episode reward. Although the dialogue length of DQN is slightly shorter than that of PA4C. the completion rate of DQN is much lower than PA4C. This is because DQN sacrifices the completion rate to obtain shorter dialogues. As a result, DQN often fails when interacting with low-confidence users. In contrast, our PA4C model is more robust to these users and can handle these uncertainties very well without causing a significant increase in dialogue length.

- **PA4C v.s. DRQN.** We can also see that PA4C significantly surpasses DRQN over 27% improvement in terms of completion rate, dialogue length, and episode reward. The performance of DRQN is even worse than DQN although LSTM is introduced into DRQN. We

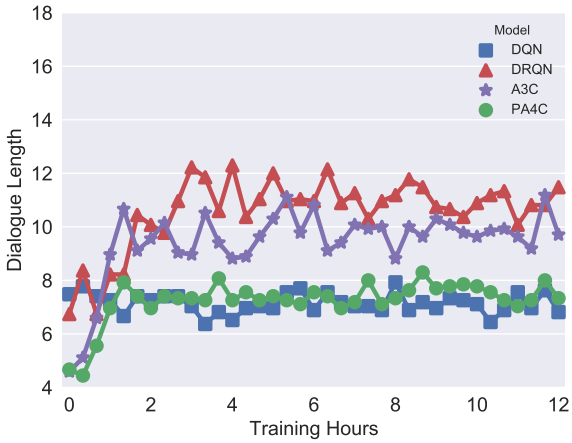


(a) Compare with other RL models: DQN, DRQN, A3C, and PA4C.

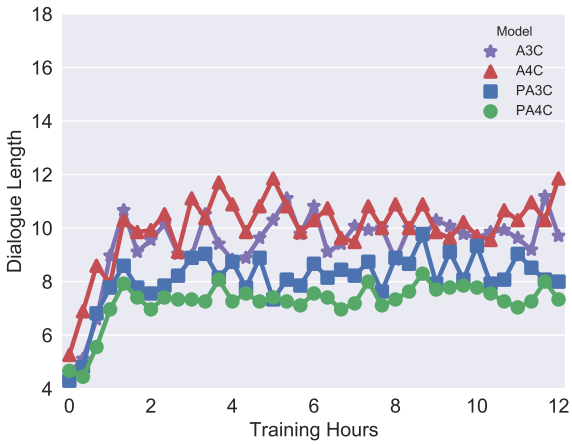
(b) Compare within proposed models: A3C, A4C, PA3C, and PA4C.

Figure 4.6: Completion rate comparison

find that simply adding LSTM does not take effect but even lowers the model performance. This is because DRQN and DQN are both value-based reinforcement learning models and are using the same way to learn policies. But DRQN has much more parameters to update, which leads to a worse performance for DRQN. Compared to DRQN, although PA4C also introduces LSTM layers, it can fully make use of LSTM layers because PA4C integrates the strength of both policy-based and value-based reinforcement learning. We also illustrate the learning curve of PA4C and DRQN in Figure 4.6a, Figure 4.7a, and Figure 4.8a. These figures show that PA4C can not only achieve a much better performance, but spend much less time to converge.



(a) Compare with other RL models: DQN, DRQN, A3C, and PA4C.



(b) Compare within proposed models: A3C, A4C, PA3C, and PA4C.

Figure 4.7: Dialogue length comparison

- **PA4C v.s. A3C.** A3C is also a state-of-the-art hybrid policy-based and value-based reinforcement learning algorithm. As shown in Figure 4.6a, Figure 4.7a, and Figure 4.8a, it outperforms both the DQN and the DRQN baselines. However, A3C still uses a single channel to output a one-hot vector for dialogue actions and does not pay particular attention to the change of dialogue state values. This means that A3C will also encounter the problem of large action spaces and zero-reward states in dialogue policy learning. In contrast, our proposed PA4C model extends from A3C, and introduces action parameterization as well as auxiliary tasks to overcome these two problems. Table 4.4, Figure 4.6a, Figure 4.7a, and Figure 4.8a also show that PA4C can surpass A3C over 10% in completion rate, dialogue length, and episode reward. Furthermore, compared with A3C, PA4C takes less time to

(a) Compare with other RL models: DQN, DRQN, A3C, and PA4C.

(b) Compare within proposed models: A3C, A4C, PA3C, and PA4C.

Figure 4.8: Episode reward comparison

converge to the best performance.

Comparison within Sub-models

To verify the effectiveness of action parameterization and auxiliary tasks, we compare PA4C with A3C, A4C, PA3C. Experimental results are shown in Table 4.4. We also plot the learning curves of these models in Figure 4.6b, Figure 4.7b, and Figure 4.8b.

- **PA3C v.s. A3C.** As Table 4.4 shows, using action parameterization (i.e., PA3C) achieves 9% improvement in completion rate, 12% in dialogue length, 9% in episode reward over the A3C baseline. But in the early stage of training, the completion rate of PA3C will meet a

slight dip and then gradually go up with the increase of training steps. It will surpass A3C after being trained over 4 hours.

- **A4C v.s. A3C.** Table 4.4 also shows that adding auxiliary tasks to A3C (i.e., A4C) can make 2% improvement in completion rate over the A3C baseline. The effect of auxiliary tasks is not as significant as action parameterization. The dialogue length and episode reward of A4C are even lower than A3C. However, it does not mean that auxiliary tasks are useless. As mentioned above, PA3C will meet a dip in the early stage of training, meaning that PA3C are not very stable. In contrast, Figure 4.6b illustrates that the learning curve of A4C keeps rising without any drop, meaning that A4C can stabilize the training process and make the result more robust.

- **PA3C + A4C v.s. A3C.** Our PA4C model is the integrated version of PA3C and A4C. As illustrated in Table 4.4, PA3C and A4C can promote together, contributing to a further improvement in the metrics of completion rate, dialogue length, and episode reward. The introducing of action parameterization (i.e., PA3C) makes 9% improvement in completion rate, 12% in dialogue length, and 9% in episode reward over the A3C baseline. Then, auxiliary tasks help PA3C continue to improve 5%, 7%, and 3% in these three metrics, respectively. Finally, PA4C can achieve 14% in completion rate, 19% in dialogue length, and 12% in episode reward over the A3C model.

4.5 Summary

In this chapter, we presented a context-uncertainty-aware policy for task-oriented dialogue systems to handle ambiguous user utterances. Task-oriented dialogue systems using this policy can adapt to various user utterances in certain dialogue contexts. We also proposed a novel reinforcement learning model named PA4C to optimize the context-uncertainty-aware policy, which can avoid a large action selection space via action parameterization and can discover valuable states via auxiliary tasks. We evaluate our model by training a dialogue system for the calendar events creation task. Experimental results show that our PA4C model outperforms the state-of-the-art models in the metrics of completion rate, dialogue length, and episode reward.

Chapter 5

Adversarial Dialogue Policies

In this chapter, we propose an adversarial dialogue policy for task-oriented dialogue systems to handle long dialogues. When training dialogue policies, many existing studies apply reinforcement learning (RL) to avoid the expensive human labour in collecting a large number of training dialogues and to improve the task completion rate. One problem in the RL training process is that RL usually requires a reward from each turn in a dialogue to effectively guide the optimal action exploration. However, it is not an easy task to obtain such rewards from real dialogue data because dialogue system users usually provide feedback only at the end of a dialogue (e.g., give a user satisfaction rating) rather than in each turn. We propose an adversarial dialogue policy to address this problem and a novel RL model named PGGAN to train the proposed policy. In the PGGAN model, a *generative adversarial net* (GAN) constantly provides intermediate rewards for every turn of the dialogue to a *policy gradient* (PG) optimized RL process. To optimally integrate PG with GAN and provide more accurate rewards, we also predict users' forthcoming responses through *Monte Carlo* simulation. We evaluate the PGGAN model extensively using real-world datasets of making restaurant reservations. The experimental results show that PGGAN significantly outperforms baselines in terms of task completion rates (up to 25% improvement).

Part of the content of this chapter has been submitted to

1. Chuandong Yin, Yu Sun, Jianzhong Qi, and Rui Zhang. PGGAN: An Adversarial Approach to Task-Oriented Dialog Policy Learning. Submitted to *the Association for the Advancement of Artificial Intelligence* (AAAI), 2018

5.1 Introduction

Task-oriented dialogue systems (dialogue systems for short in the rest of this chapter) are an important type of artificial intelligence (AI) applications. When interacting with users, dialogue systems help users complete tasks of specific domains. Examples of such dialogue systems are Apple Siri and Microsoft Cortana. However, many existing dialogue systems can only handle short dialogues that usually can be completed less than 3 dialogue turns such as checking the weather or setting an alarm. They have a very limited ability to conduct long dialogues that may take over 20 dialogue turns such as making restaurant reservations. Figure 5.1 shows an example dialogue for restaurant reservation that has 23 turns. We aim to build a dialogue system that can complete such long dialogues.

A traditional approach to learn a dialogue policy is supervised learning (SL). However, SL is often data-hungry and requires a large number of training dialogues to cover the highly diverse user utterances. Therefore, reinforcement learning (RL) methods are proposed to train a dialogue policy with limited data [98, 19]. The RL methods use a user simulator to mimic human responses and to interact with a dialogue system to be trained. The user simulator and the system correspond to the *environment* and *agent* in RL, respectively. By repeatedly interacting with the simulator, the system learns to handle different user inputs and eventually becomes ready to handle tasks of real users. To further illustrate how this training process works, consider training a dialogue system for making restaurant reservations. In each training task, the simulator fetches a random restaurant preference from a pre-built preference database. It then asks the system to book a restaurant that suits this preference. The system tries to complete the task by requesting booking preferences (e.g., the preferred type of cuisine) from the simulator. The simulator responds with the requested booking preferences piece by piece. At the end of the dialogue, the simulator checks whether the system successfully books a restaurant that suits its preferences and then gives the system a *global* positive or negative reward as the feedback of the dialogue quality. The system can get improved according to the global reward from the simulator.

Although using user simulators can help reduce the number of training dialogues. this approach also has a large limitation that the global reward is only available at the end of a dialogue (i.e., after the last dialogue turn). In other words, there are no *local* rewards available to evaluate the quality of the response in each intermediate dialogue turn. This is because real users will not

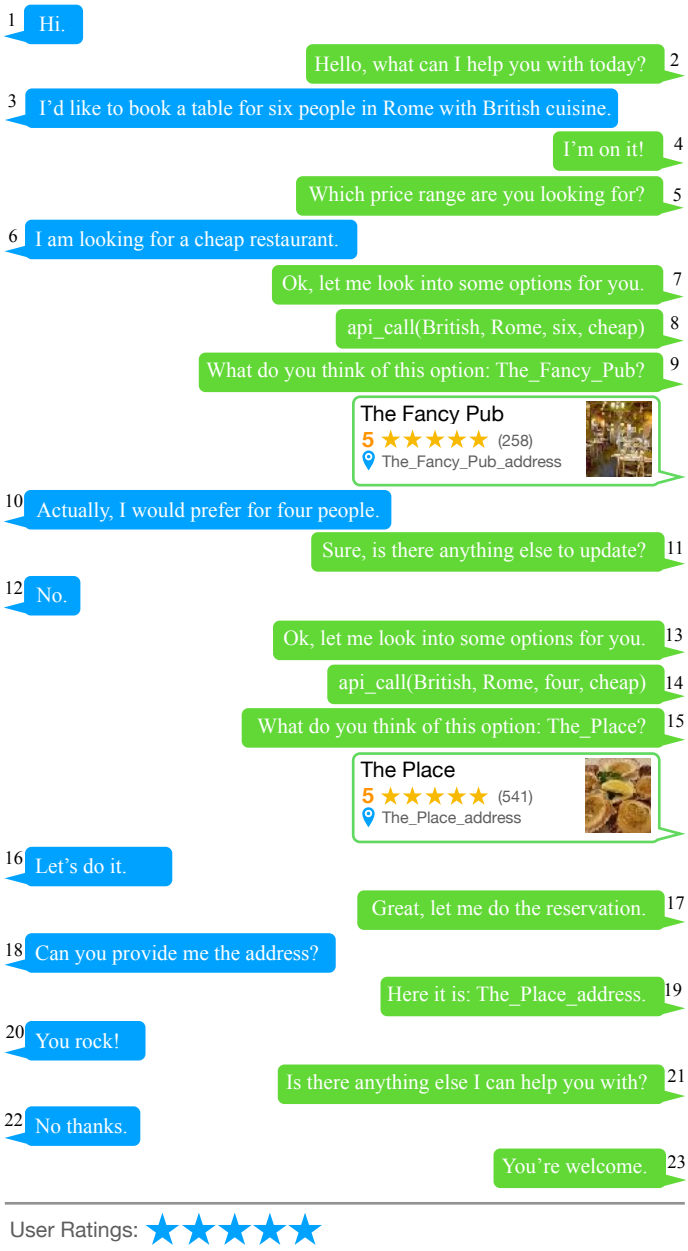


Figure 5.1: An example dialogue generated by PGGAN

explicitly rate the experience of each dialogue turn when using dialogue systems. We call this problem the *sparse rewards* problem, which makes it difficult for dialogue systems to improve their responses on a turn-by-turn level and may eventually cause the failure of the task due to lack of guidance in the response (i.e., action) exploration. Existing studies bypass the sparse rewards

problem by *reward shaping* [98, 19]. They assign a small fixed negative local reward (e.g., -0.1) for intermediate turns and a large global reward (e.g., -1 or 1) for the entire dialogue. Such an approach does not alleviate the problem because the small negative local reward does not represent the quality of system responses. It only encourages the system to finish the dialogue as soon as possible because of the large global reward. Besides, dialogue systems supported by knowledge bases (KB) uses the number of entities matched in KB as pseudo rewards [104]. However, the KB-associated pseudo rewards only force the dialog agent to match more entities, even if these entities are not related to user preferences. Moreover, such pseudo rewards are not available in every turn. Only when the dialog agent queries KB, the user simulator is able to give the reward that is related to the number of matched entities. This often leads to failing users' tasks, especially when a user changes her mind during a dialogue (e.g., another type of preferred cuisine just comes into mind), in which case a longer dialogue is needed to clarify the new booking preference.

We aim to overcome the sparse rewards problem and achieve a dialogue system that can not only complete a user's task but also produce valid responses in each dialogue turn. Here, "*valid*" means that the responses generated by the system (i.e., *generated responses*) resemble those from real human (i.e., *human responses*) and make sense in the dialogue context. For example, if a user says "hello", a valid response can be "hello, how can I help you?", while an invalid response may be "is there anything else to update?". Invalid responses may discourage a user to continue using dialogue systems, which should be avoided. While it might seem easy for a human user to distinguish an invalid response from a valid one, this is difficult for the user simulator because it requires the user simulator to learn the semantic meaning of the responses. As a result, it is a challenging problem for the user simulator to generate a local reward according to the validity of a system response. In this study, we address this problem via a novel reward learning approach to help the user simulator differentiate these two types of responses and to learn different rewards for them, which overcomes the sparse rewards problem and improves the user experience of dialogue systems in each dialogue turn.

Our reward learning approach is inspired by *generative adversarial nets* (GAN) [22]. We design a *discriminator* to differentiate generated responses from human responses. It outputs a local reward between -0.1 and 0.1 representing how well a generated response resembles a human response. Then we feed the local reward into the system which is now treated as a *generator*. The

generator is optimized towards obtaining larger local rewards: it needs to generate more human-like responses to confuse the discriminator. In contrast, the discriminator is optimized towards differentiating generated responses from human responses as accurate as possible, leading to more accurate local rewards. When the discriminator cannot distinguish the generated responses, it means that the generator (i.e., system) is responding like a human. This process resembles a minimax game, but we expect the generator to win the game. Therefore, such a policy is called an *adversarial dialogue policy*.

Moreover, to avoid overly optimizing the adversarial dialogue policy towards a larger local reward in a dialogue turn t, we take into account the rewards of all future turns $t+1, ..., T$ (T represents the last turn) via *Monte Carlo* simulation. In this way, the reward considered in each turn consists of both a local reward to indicate whether the generated response resembles a human response, and a global reward to indicate the predicted completion or failure of the dialogue given the currently generated response, making the generated responses much more reliable. Thus, our adversarial dialogue policy is able to handle long dialogues.

Overall, this chapter makes the following contributions:

- We build an adversarial dialogue policy that can handle long dialogues that may take more than 20 turns to complete. In particular, we propose a novel reinforcement learning model named PGGAN by systematically integrating policy gradient with GAN to train such a policy.

- We train a discriminator to differentiate generated responses from human responses and to provide additional turn-based rewards for reinforcement learning models, which can overcome the sparse rewards problem in dialogue policy learning.

- When generating responses in dialogue turn t, we take into account both the local reward for the current turn and the global reward for the entire dialogue through *Monte Carlo* simulation from turn $t+1$ to turn T. This makes the generated responses much more reliable.

- We evaluate the proposed PGGAN model by training a task-oriented dialogue system to make restaurant reservations using the adversarial dialogue policy. The experimental results show that PGGAN can significantly reduce the number of training dialogues and consistently improve the task completion rate compared with the baseline models.

5.2 Preliminaries

In this section, we briefly discuss the basics of policy gradient and generative adversarial nets.

5.2.1 Policy Gradient

Policy gradient algorithms are widely used to find the optimal policy (i.e., the action to be taken given an environment state) in reinforcement learning (RL) models. RL is modelled as a *Markov decision process* (MDP), which can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ denotes the state space; $\mathcal{A}$ denotes the action space; $\mathcal{P}$ denotes the transition probability $P(s_{t+1}|s_t, a_t)$ from state s_t to s_{t+1} with action a_t (where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$); $\mathcal{R}$ denotes the reward function $\mathcal{R}(s_t, a_t)$ by taking action a_t on state s_t; $\gamma \in (0, 1]$ denotes the discount factor. The *return* $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the cumulative discounted reward from step t onwards, where $r_t = \mathcal{R}(s_t, a_t)$. An action-value function $Q^{\pi_\theta}(s, a) = \mathbb{E}_{\pi_\theta}[R_t|s_t = s, a_t = a]$ computes the expected return of taking action a_t on s_t, where π_θ is the *policy* representing how to choose the action on state s_t. The goal of RL is to obtain an optimal policy π_{θ^*} which can maximize the value of $Q^{\pi_\theta}(s, a)$.

Both value-based and policy-based RL can be used to find the optimal policy π_{θ^*}. Value-based RL, such as *Q-learning* [88, 58], first computes the optimal action-value function $Q^*(s, a)$ where:

$$Q^*(s, a) = \max_{\pi_\theta} \mathbb{E}_{\pi_\theta}[R_t|s_t{=}s, a_t{=}a] \tag{5.1}$$

and then computes the optimal policy π_{θ^*} according to Q^*.

$$\pi_{\theta^*} = \operatorname*{argmax}_a Q^*(s, a) \tag{5.2}$$

The *Bellman equation* [88] is widely used to search for the optimal action-value function Q^* by recursively obtaining the maximized action-value on each state:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})|s_t, a_t\right] \tag{5.3}$$

Instead of directly optimizing the policy, value-based RL computes the optimal action-value function $Q^*(s, a)$ and optimal policy π_{θ^*} separately, which may cause biases in the action selection [59, 35]. To overcome these biases, *policy gradient* is proposed to directly optimize the policy. The rationale is to update the policy parameters θ in the direction of the performance gradients $\nabla_\theta J(\theta)$ [77]. For a stochastic policy, θ can be optimized with:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right] \tag{5.4}$$

where $\pi_\theta(a|s)$ is the policy that gives the probability of taking action a on state s and $Q^{\pi_\theta}(s,a)$ is the action-value function. Note that policy gradient does not fix the sparse rewards problem because it does not introduce any local rewards.

5.2.2 GAN

Generative adversarial nets (GANs) are used to train a generative model via a minimax game against a discriminative model. GAN consists of two modules: (1) a generator G to capture the data distribution, and (2) a discriminator D to estimate the probability that a sample comes from the data [22, 100, 50]. G is trained to generate fake samples that confuse D, while D is optimized to differentiate fake samples from real data. This process forms a minimax game between G and D with value function $V(G,D)$.

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\Big[\log D(x)\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[\log\left(1-D(G(z))\right)\Big] \tag{5.5}$$

where $D(x)$ is the probability that sample x comes from real data; $G(z)$ is a fake sample generated with an input noise z; $p_z(z)$ is a prior noise distribution. When training GAN, we update G to minimize $\log\left(1-D(G(z))\right)$ and D to maximize $\log D(x)$.

Note that the generator G in its vanilla form can only generate data in a continuous space, such as images, while the action space in policy learning for task-oriented dialogue systems is discrete. Therefore, it cannot be directly applied to learn dialogue policies.

5.3 PGGAN Model

In this section, we present our model named PGGAN to solve the sparse rewards problem in dialogue policy learning.

5.3.1 Model Overview

Figure 5.2 illustrates the structure of PGGAN. PGGAN consists of three parts: (1) an agent (forming the *generator* together with a user simulator which will be detailed in Section 5.4.1), (2) a reward function (*discriminator*), and (3) a Monte Carlo simulation process. The agent keeps dialoguing with the user simulator until the end of a dialogue. This process produces a sequence of

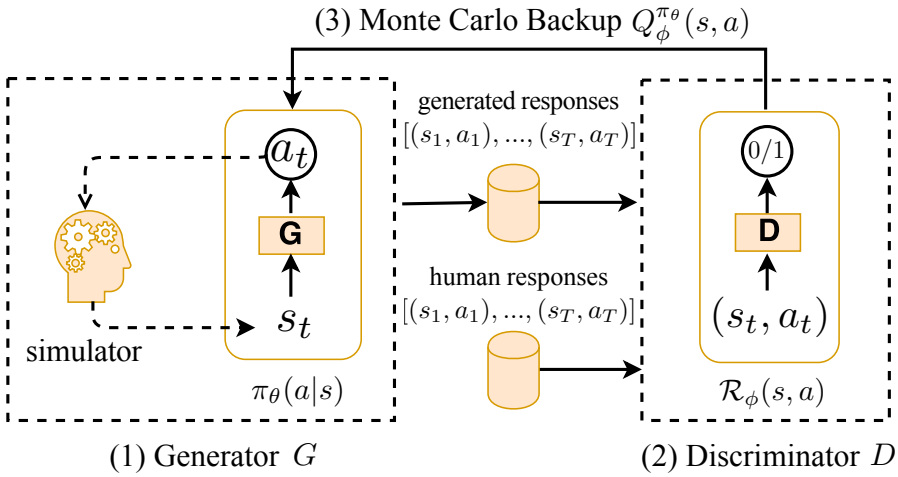


Figure 5.2: Overview of PGGAN

(s,a) pairs, e.g., $[(s_1,a_1),(s_2,a_2)...,(s_T,a_T)]$, where s denotes the dialogue context, a denotes the response chosen by the generator, and T denotes the last dialogue turn. We save this sequence in a buffer called *generated responses*. Next, we feed this sequence into the discriminator to obtain a reward for each (s,a) pair. We then backup these rewards into $Q(s,a)$ from the last turn T to the current turn t (i.e., Monte Carlo Backup) and use $Q(s,a)$ to improve the generator.

When training the discriminator, we also parse the dialogues collected from real-world interactions into sequences of (s,a) pairs and save them in another buffer called *human responses*. We train the discriminator using generated responses and human responses, and encourage the discriminator to differentiate these two types of responses to provide more accurate rewards for the generator. On the other hand, so as to obtain larger rewards, the generator tries to generate responses to confuse the discriminator. This process resembles a minimax game but we expect the generator to win the game, meaning that the generated responses are supposed to be as good as human responses at the end of the training process. Therefore, we define the overall optimization objective to be:

$$J^{G^*,D^*} = \min_\theta \max_\phi \mathbb{E}_{a\sim\pi_{\text{true}}(a|s)}\left[\log \sigma(d_\phi(a|s))\right] + \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(1-\sigma(d_\phi(a|s))\right)\right] \quad (5.6)$$

where $\pi_\theta(a|s)$ denotes the generator G that outputs a probability distribution of responses; $\sigma(d_\phi(a|s))$ denotes the discriminator D that predicts a score between 0 and 1 to estimate the similarity between generated responses and human responses.

Note that this optimization function differs from the original GAN optimization function

(Eq. 5.5) in that $\pi_\theta(a|s)$ in Eq. 5.6 can output a discrete space by policy gradient and Monte Carlo simulation, while $p_z(z)$ in Eq. 5.5 can only output a continuous space. Such a difference enables the generator to handle *Markov decision process* by reinforcement learning, which can overcome the sparse rewards problem in policy learning for task-oriented dialogue systems.

5.3.2 Reward Function (Discriminator)

As mentioned before, when using RL to train dialogue policies, most models will encounter the problem of sparse rewards. This is because a dialogue system user will only give a global reward as the feedback after the agent finishes the dialogue and there are no local rewards in intermediate dialogue turns to evaluate the quality of each agent response. As a result, the dialogue agent may take much longer time to search for the optimal response which may lead the agent to get trapped into local optima.

For the sake of obtaining local rewards, we plan to build a reward function to provide rewards for the dialogue agent in each intermediate dialogue turn. A traditional way to build such a reward function is to train a regression model that outputs a score for each generated response. However, to train such a model, we have to spend a lot of human effort in labeling a score for each available response. In spite of doing this, these labeled scores may cause many variances due to the subjectivity of different people, which will make the reward function less reliable.

To address this issue, we adapt the idea of GAN to build the reward function. In particular, we train a discriminator to provide turn-by-turn rewards. Unlike traditional regression models that require a large number of manually labeled scores for responses, the discriminator just needs moderate example responses from real human (i.e., human responses) as positive samples. On the other hand, the interaction between the agent and the user simulator will produce a series of responses (i.e., generated responses), which will be exploited by the discriminator as negative samples. The discriminator accepts generated responses and human responses as the inputs and is trained to differentiate them. It will output a similarity score between 0 and 1 to represent how a generated response resembles a human response. Such a similarity score will be treated as the local reward in each intermediate dialogue turn. In this way, we can save a lot of human labours in labeling scores for each response.

When building the discriminator, we therefore use the following *sigmoid* function that ranges

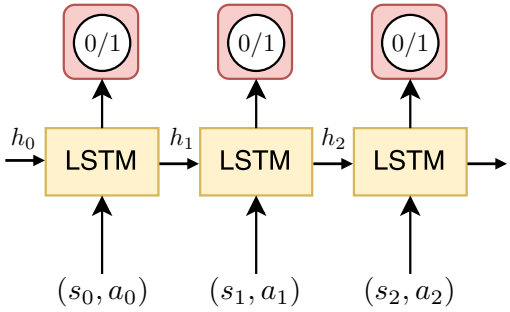


Figure 5.3: The model structure of the discriminator.

from 0 to 1:

$$\sigma(d_\phi(a|s)) = \frac{\exp(d_\phi(s,a))}{\exp(d_\phi(s,a)) + 1} \tag{5.7}$$

where σ is the sigmoid function and $d_\phi(s,a)$ is a function that accepts a dialogue context (i.e., state) s and a response (i.e., action) a as its input. As shown in Figure 5.3, we use a recurrent neural network (RNN) as $d_\phi(s,a)$ in our model, because dialogue policy learning is a sequential decision-making process and RNN can capture the hidden information of historical dialogue turns. The objective is to maximize the log-likelihood of correctly differentiating generated responses from human responses, which is formulated as follows:

$$\phi^* = \arg\max_\phi \mathbb{E}_{a\sim\pi_{\text{true}}(a|s)}\left[\log\left(\sigma(d_\phi(s,a))\right)\right] + \mathbb{E}_{a\sim\pi_{\theta^*}(a|s)}\left[\log\left(1-\sigma(d_\phi(s,a))\right)\right] \tag{5.8}$$

where $\pi_{\theta^*}(a|s)$ is the current optimal policy for the generator.

5.3.3 Agent (Generator)

As shown in Figure 5.4, the dialogue agent (i.e., generator) aims to produce a proper response a given a dialogue context (state) s. It can be expressed by $\pi_\theta(a|s)$, where π denotes the policy to learn and θ denotes the parameters of the policy π. Intuitively, the generator needs to generate responses which are as good as human responses so that it can obtain larger rewards from the discriminator, meaning that the generator can perform like a real human in a task-oriented dialogue system. Therefore, we train the generator to minimize $\log\left(1-\sigma(d_\phi(s,a))\right)$ as defined in Eq. 5.6.

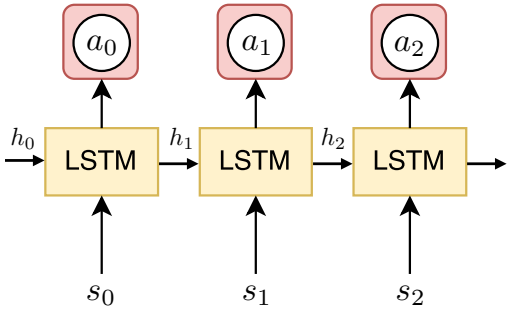


Figure 5.4: The model structure of the generator.

The objective of the generator can be formulated in Eq. 5.9:

$$\begin{aligned}\theta^* &= \arg\min_\theta \mathbb{E}_{a\sim\pi_{\text{true}}(a|s)}\left[\log\left(\sigma(d_\phi(s,a))\right)\right] + \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(1-\sigma(d_\phi(s,a))\right)\right] \\ &= \arg\min_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(1-\frac{\exp(d_\phi(s,a))}{\exp(d_\phi(s,a))+1}\right)\right] \\ &= \arg\max_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(1+\exp(d_\phi(s,a))\right)\right]\end{aligned} \tag{5.9}$$

However, we cannot straightforwardly use gradient descent techniques to optimize the generator. This is because the responses a in the user simulator are discrete whereas the vanilla GAN formulation in Eq. 5.5 is only able to handle a continuous space. Therefore, we propose to use *policy gradient* to overcome this problem. As we mentioned in Section 5.2.1, policy gradient can define a stochastic policy $\pi_\theta(a|s)$ to handle discrete actions. By doing so, the generator can be directly optimized by gradient descent. We define a reward function $\mathcal{R}_\phi(s,a)$ to represent the output of the discriminator:

$$\mathcal{R}_\phi(s,a) = \log\left(1+\exp(d_\phi(s,a))\right) \tag{5.10}$$

Then, we replace the term $\log\left(1+\exp(d_\phi(s,a))\right)$ in Eq. 5.9 with $\mathcal{R}_\phi(s,a)$ and can obtain a new objective as follows:

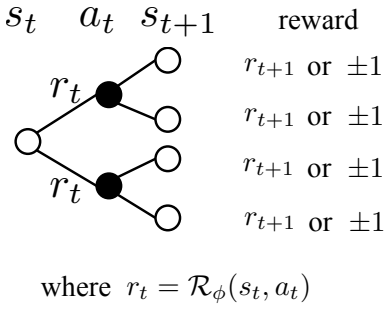
$$\theta^* = \arg\max_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\mathcal{R}_\phi(s,a)\right] \tag{5.11}$$

In this way, Eq. 5.11 can be easily optimized by gradient descent. The derivative of the generator

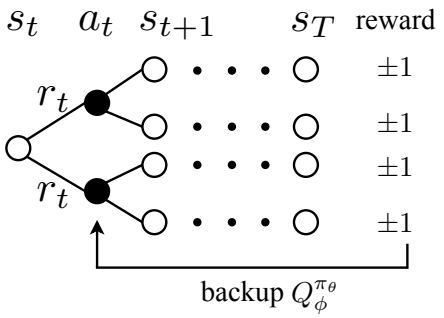
can be obtained according to:

$$\begin{aligned}\nabla_\theta J^G(\theta) &= \nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta(a|s)}\Big[\mathcal{R}_\phi(s,a)\Big] \\ &= \sum_{i=1}^{K} \nabla_\theta \pi_\theta(a_i|s)\mathcal{R}_\phi(s,a_i) \\ &= \sum_{i=1}^{K} \pi_\theta(a_i|s)\nabla_\theta \log\, \pi_\theta(a_i|s)\mathcal{R}_\phi(s,a_i) \\ &= \mathbb{E}_{a\sim\pi_\theta(a|s)}\Big[\nabla_\theta \log\, \pi_\theta(a|s)\mathcal{R}_\phi(s,a)\Big]\end{aligned} \tag{5.12}$$

where K is the number of possible agent responses and $\mathcal{R}_\phi(s,a)$ is the immediate reward given by the discriminator.



(a) Non-MC



(b) MC

Figure 5.5: Non-MC vs MC when computing the action-value.

5.3.4 Monte Carlo (MC) Simulation

Nevertheless, simply combining policy gradient and GAN such as Eq. 5.12 has a large drawback. Actually, Eq. 5.12 just represents a one-step MDP, meaning that it only considers the reward in the next one step given by the discriminator. As shown in Figure 5.5a, when generating responses on state s_t, Eq. 5.12 only computes the immediate reward $r_t = \mathcal{R}_\phi(s_t, a_t)$ (i.e., local rewards) but ignores the influence of subsequent states $(s_{t+1}, ..., s_T)$ (i.e., global rewards). If we directly apply such a formulation to optimize dialogue policies, the agent will become shortsighted when producing responses, leading to failing user tasks. This is mainly because the agent only manages to respond like a human so as to obtain larger local rewards from the discriminator but forgets to complete user tasks. Therefore, we need to find a balance between mimicking human responses and completing user tasks.

As we mentioned before, dialogue policy learning is a sequential decision-making process, meaning that the subsequent responses will be affected by the choice of responses in previous turns. Considering this scenario, we thus propose to adapt Monte Carlo simulation to train the generator to find the balance. In this way, the agent can interact with the user simulator until the end of the dialogue and then backpropagate the rewards from the last turn to the current turn, making the final reward (i.e., action-value) in each dialogue turn incorporate both local rewards and global rewards. Figure 5.5b demonstrates this process, to evaluate the action-value of choosing response a_t in dialogue state s_t, we run the Monte Carlo simulation with a roll-out policy π_θ from s_t to s_T (recall that T is the last dialogue turn) and trace back the action-value from s_T to s_t. We define function $Q_\phi^{\pi_\theta}(s, a)$ to represent the action-value:

$$Q_\phi^{\pi_\theta}(s_t, a_t) = \begin{cases} \mathcal{R}_\phi(s_t, a_t) + \gamma Q_\phi^{\pi_\theta}(s_{t+1}, a_{t+1}), & t < T \\ \pm 1, & t = T \end{cases} \tag{5.13}$$

Replacing $\mathcal{R}_\phi(s, a)$ in Eq. 5.12 with $Q_\phi^{\pi_\theta}(s, a)$, we can obtain the final generator derivative:

$$\nabla_\theta J^G(\theta) = \mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q_\phi^{\pi_\theta}(s, a) \right] \tag{5.14}$$

Algorithm 4 summarizes the overall training procedure.

Algorithm 4 Training of PGGAN

Pre-train G with human dialogues.
for g-steps **do**
    Initialize a user simulator with a state s
    Initialize a response *buffer* = []
    while s is not the last turn **do**
        G generates a response a with $\pi_\theta(a|s)$
        Respond the simulator with a and get a new state s'
        D predicts a reward r by feeding with (s, a)
        Save (s, a, r) into *buffer*
        $s \leftarrow s'$
    Compute $Q_\phi^{\pi_\theta}$ by Monte Carlo backup via Eq. 5.13.
    Update G with *buffer* via Eq. 5.14.
for d-steps **do**
    Run G to generate responses and save into *buffer_neg*
    Sample human responses and save into *buffer_pos*
    Update D with *buffer_neg* and *buffer_pos* via Eq. 5.8.

5.4 Experiments

We evaluate the proposed PGGAN model by training a task-oriented dialogue system on the task of making restaurant reservations using the adversarial dialogue policy. We use the open-source dataset *bAbI* ("Task5-OOV" and "Task6") released by Bordes and Weston [11]. Each task consists of a training set, a validation set, and a test set, each of which contains $1{,}000$ dialogues. We also design a user simulator for this dataset to interact with RL models. In our experiments, we use *completion rate* as the evaluation metric [65, 62] for all models, where "completion" means that a user successfully books a restaurant via the dialogue system and can be given by the user simulator.

5.4.1 User Simulator

To train a dialogue policy through RL approaches, we need a user simulator to simulate human behaviour and to interact with the dialogue system. We use an open-source user simulator proposed by Li et al. [44], which is built on the domain of booking movies. Then we adapt their simulator to the domain of making restaurant reservations. Overall, our user simulator is able to (1) inform keywords (e.g., the preferred type of cuisine) to the system, (2) update the previous keywords, (3) accept or reject the recommendations from the system, (4) request the details of the finalized

reservation from the system (e.g., restaurant address). Furthermore, the input and output in our user simulator are natural language sentences. We map each dialogue action to a unique natural language template.

The simulation process is briefly illustrated as below. At the beginning of each simulated dialogue, we randomly sample (1) attribute-value pairs $\mathbf{U}_{\text{inf}} = \{(k_1^{\text{inf}}, v_1^{\text{inf}}), ..., (k_n^{\text{inf}}, v_n^{\text{inf}})\}$ that the simulator will inform to the system, such as $\{$("cuisine", "Korean")$\}$, and (2) attribute-value pairs $\mathbf{U}_{\text{req}} = \{(k_1^{\text{req}}, v_1^{\text{req}}), ..., (k_n^{\text{req}}, v_n^{\text{req}})\}$ that the simulator will request from the system, such as $\{$("phone", "*")$\}$, where "*" is a placeholder that needs to be filled by the system. During a dialogue simulation, $\mathbf{U}_{\text{inf}}$ and $\mathbf{U}_{\text{req}}$ are known to the simulator but hidden from the system. In other words, $\mathbf{U}_{\text{inf}}$ and $\mathbf{U}_{\text{req}}$ are the *ground truth*. The task of the dialogue system is to recognize the content of $\mathbf{U}_{\text{inf}}$ first, and then to correctly fill the placeholder "*" in $\mathbf{U}_{\text{req}}$ through the interaction with the simulator, i.e., to complete a restaurant reservation task. Note that in each dialogue turn t, the attribute-value pairs ($\mathbf{U}_{\text{inf}}$ and $\mathbf{U}_{\text{req}}$) will be transformed to natural language sentences by a NLG component provided by Li et al. [44]. Therefore, the models in our experiments are actually trained in an end-to-end manner.

5.4.2 Baseline Models

We compare PGGAN with a retrieval-based model, a supervised learning model, and a vanilla RL model.

- **Retrieval-based.** This model computes the cosine similarities between the word embeddings of the last user response and all possible user responses in the training set. Then it selects the system response which is mapped by the user response with the largest similarity to respond users.

- **Supervised Learning (SL).** This model consists of an LSTM layer and a fully-connected layer, which outputs meta system actions. We sample context-response pairs from human dialogues in the training set. Then, we feed these pairs into the model and train the model using the softmax cross entropy loss.

- **Policy Gradient (PG).** The structure of this model is the same as the SL model. We first pre-train the PG model with a subset of the human dialogues, and then continuously optimize

Table 5.1: Completion rate comparison when training with different numbers of dialogues on "Task5-OOV". "AE" means action exploration, "MC" means Monte Carlo simulation, and "—" means not applicable. The result is averaged over 5 runs.

	Setting			Training dialogues								
	Model	AE	MC	10	30	50	80	100	300	500	800	1000
Baselines	Retrieval	—	—	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	SL	—	—	9.2%	22.8%	58%	66.2%	68.0%	74.8%	80.2%	82.6%	83.4%
	PG	✗	✗	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		✓	✗	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		✗	✓	9.2%	21.6%	65.4%	65.2%	75.4%	73.4%	81.2%	81.8%	84.0%
		✓	✓	9.0%	24.6%	73.2%	82.8%	86.8%	83.9%	88.0 %	88.4%	87.8%
Ours	PGGAN	✗	✗	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		✓	✗	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		✗	✓	**9.6%**	30.6%	67.4%	73.4%	78.2%	78.5%	86.8%	84.4%	86.4%
		✓	✓	7.6%	**50.2%**	**91.2%**	**94.6%**	**95.2%**	**95.6%**	**95.2%**	**95.8%**	**96.6%**

the model with action exploration via the user simulator. We set the rewards of intermediate turns and the last turn to be -0.1 and 1.0 [19], respectively. We implement four variants of this model by combining the following options: (1) with or without action exploration (AE), and (2) with or without Monte Carlo simulation (MC).

- **PGGAN.** We implement the proposed model by making the generator the same as PG. Besides, we introduce a discriminator which consists of an LSTM layer followed by a fully-connected layer and outputs the reward for generated responses. In the action exploration stage, we feed all context-response pairs into the discriminator to obtain corresponding rewards (we scale the rewards between -0.1 and 0.1). We also implement four variants of PGGAN like PG: (1) with or without action exploration (AE), and (2) with or without Monte Carlo simulation (MC).

5.4.3 Hyperparameters

The input of all models is a 53-dimensional vector (a 50-dimensional average GloVe [61] word embedding of the last system response and the current user response, and a 3-dimensional one-hot vector indicating the number of search results). All LSTM layers have 128 hidden units. The user simulator has 17 meta actions. All models are trained using the RMSProp optimizer with a learning rate of 0.001 and a weight decay of 0.99. We set the batch size to 16, the dropout rate to 0.5, and the discounted factor γ of RL models to 0.99. We use early stopping in training process

Table 5.2: Dialogue length comparison when training with different numbers of dialogues on "Task5-OOV". A dialogue turn includes a user response and a system response. The result is averaged over 5 runs.

	Setting			Training dialogues								
	Model	AE	MC	10	30	50	80	100	300	500	800	1000
Baselines	Retrieval	—	—	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000
	SL	—	—	7.224	10.244	12.792	13.268	13.852	14.266	15.762	15.548	16.536
	PG	✗	✗	2.760	2.770	2.894	2.924	2.732	2.078	1.796	1.592	1.460
		✓	✗	2.914	2.858	2.806	2.754	2.800	2.106	2.152	3.824	2.954
		✗	✓	6.882	9.548	13.336	13.598	14.710	14.347	15.506	16.164	15.815
		✓	✓	8.722	12.816	13.648	16.820	17.806	16.943	17.754	17.770	17.736
Ours	PGGAN	✗	✗	2.988	2.946	2.800	2.606	2.496	2.446	1.810	1.540	1.482
		✓	✗	2.938	2.720	2.874	2.712	2.546	2.444	2.034	4.570	2.954
		✗	✓	8.474	11.494	13.790	15.936	14.686	15.015	16.638	15.760	16.050
		✓	✓	7.828	14.174	15.478	16.132	16.388	16.902	16.514	16.554	16.492

and set the early stopping epoch to 10.

5.4.4 Results

Table 5.1 and Table 5.2 summarize the completion rates and dialogue length of different models. The comparisons are detailed as follows.

Comparison with Retrieval-based Model

We firstly compare our proposed PGGAN model with a retrieval-based model. As Table 5.1 shows, the retrieval-based model cannot complete even one dialogue (0.0% completion rate), while PGGAN (with AE and MC) can achieve over 95% completion rates. This is because the retrieval-based model does not consider the dialogue context, leading to repeating the most similar response until reaching a pre-defined maximum dialogue length (30 turns in our experiments). In contrast, PGGAN can learn the hidden correlation between a system response and a dialogue context, which is a determinant factor to complete a dialogue. Table 5.2 shows the details of the length of dialogues generated by the retrieval-based model and the PGGAN model. We find that although the retrieval-based model takes 30 dialogue turns, it cannot complete even one dialogue. While PGGAN (with AE and MC) only takes less than 17 dialogue turns and can reach over 96% completion rates. It means that handling long dialogues is a very challenging problem and the retrieval-based model does not have the ability to solve it. Fortunately, our proposed PGGAN model can well complete those long dialogues that need multiple user-system interactions.

Comparison with Supervised Learning Model

We also compare the PGGAN model with a supervised learning model. From Table 5.1, we can see that the SL model only reaches 83.4% completion rates despite training with $1{,}000$ human dialogues, while PGGAN (with AE and MC) can achieve over 95% completion rates with only 100 training dialogues. Besides, Table 5.2 shows that PGGAN does not cause a significant increase of the dialogue length: 16.388 for PGGAN using 100 training dialogues v.s. 16.536 for SL using $1{,}000$ training dialogues. It shows that PGGAN can considerably reduce the required number of training dialogues from $1{,}000$ to 100, which is an important improvement for dialogue policy learning. As we mentioned before, it is labour-intensive to collected a large number of expert dialogues required by machine learning models. In our experiments, we utilize a user simulator to interact with the PGGAN model, which can explore many responses uncovered by the dataset. By doing so, we can save a lot of human labours in data collection.

Comparison with Reinforcement Learning Model

To justify the effectiveness of introducing the discriminator (i.e., the adversarial dialogue policy), we compare PGGAN (with AE and MC) with PG (with AE and MC) in Table 5.1. With the increase of the number of training dialogues, PGGAN can achieve $95\%+$ completion rates with only 100 training dialogues, while PG just reaches at most 88% despite training with $1{,}000$ dialogues. This is because PG uses a fixed local reward -0.1 for all intermediate dialogue turns. Such a reward cannot evaluate the quality of the response generated in each dialogue turn because the dialogue system can obtain useful rewards only at the end of the dialogue, such as $+1.0$ or -1.0 to indicate whether a task is completed or not. This will cause a significant delay in choosing the next action. In contrast, the PGGAN model utilizes a discriminator to provide meaningful rewards, which can represent whether the generated responses resemble human responses. In this way, the dialogue system can immediately get to know the quality of each generated response without any delay and therefore can make a robust decision on the next action, which contributes a lot to the task completion rate.

Comparison across Variants of PGGAN

We also compare different variants of the proposed PGGAN model to justify the effectiveness of action exploration (AE) and Monte-Carlo (MC) simulation.

- **AE v.s. Non-AE.** From Table 5.1, we can see that the introducing of action exploration significantly improves the completion rate. When training with $1{,}000$ dialogues, the completion rate of PGGAN using MC can be improved from 86.4% (without AE) to 96.6% (with AE). This is because action exploration can discover more responses not covered by the dataset. Figure 5.6 and Figure 5.7 show two dialogues generated by the PGGAN model. Although these two dialogues are not covered by the training set, our PGGAN model can still handle these unseen user inputs. However, for the model without action exploration, it usually fails when handling unseen user inputs. This proves that action exploration can significantly improve the task completion rate. Moreover, Table 5.1 also shows that action exploration can help to reduce the number of training dialogues, which can save a lot of human effort in data collection.

- **MC v.s. Non-MC.** From Table 5.1, we can find that neither PGGAN nor PG works without Monte Carlo simulation (0.0% completion rate), because they only take into account the local rewards that indicate whether the generated response resembles a human response in each turn, but ignore the global rewards to indicate the success or failure of the task. In comparison, RL models (PGGAN and PG) with MC can incorporate both local and global rewards, which makes the response generated in each dialogue turn much more reliable.

1 usr: Hi.
2 bot: Hello, what can I help you with today?
3 usr: Can you make a restaurant reservation with Korean cuisine in a cheap price range?
4 bot: I'm on it!
5 bot: Where should it be?
6 usr: In Seoul.
7 bot: how many people would be in your party?
8 usr: We will be four people.
9 bot: Ok, let me look into some options for you.
10 bot: api_call(Korean, Seoul, four, cheap)
…..list of restaurants…..
11 bot: What do you think of this option: resto_seoul_cheap_korean_2stars?
12 usr: Let's do it.
13 bot: Great let me do the reservation.
14 usr: Can you provide me the address?
15 bot: Here it is: resto_seoul_cheap_korean_2stars_address.
16 usr: Can you provide me the phone?
17 bot: Here it is: resto_seoul_cheap_korean_2stars_phone.
18 usr: You rock!
19 bot: Is there anything else I can help you with?
20 usr: No thanks.
21 bot: You're welcome.

Figure 5.6: An example dialogue generated by PGGAN. The user did not update her intents.

1 usr: Hi.
2 bot: Hello, what can I help you with today?
3 usr: May I have a table for six people with Thai food.
4 bot: I'm on it!
5 bot: Where should it be?
6 usr: In Tokyo.
7 bot: Which price range are looking for?
8 usr: Expensive
9 bot: Ok, let me look into some options for you.
10 bot: api_call(Thai, Tokyo, six, expensive)
…..list of restaurants…..
11 bot: What do you think of this option:
resto_tokyo_expensive_thai_3stars?
12 usr: Instead could it be for four people?
13 bot: Sure, is there anything else to update?
14 usr: No.
15 bot: Ok, let me look into some options for you.
16 bot: api_call(Thai, Tokyo, four, expensive)
…..list of restaurants…..
17 bot: What do you think of this option:
resto_tokyo_expensive_thai_8stars?
18 usr: Let's do it.
19 bot: Great let me do the reservation.
20 usr: Can you provide me the address?
21 bot: Here it is: resto_tokyo_expensive_thai_8stars_address.
22 usr: Thank you.
23 bot: Is there anything else I can help you with?
24 usr: No thanks.
25 bot: You're welcome.

Figure 5.7: An example dialogue generated by PGGAN. The user updated her intents.

Table 5.3: Comparison with existing work. "—" means not applicable. The result is averaged over 5 runs.

		Task5-OOV			Task6		
		Turn Acc.	Dialog Acc.	CR	Turn Acc.	Dialog Acc.	CR
SL	Bordes and Weston [11]	77.7%	0.0%	32.4%	41.1%	0.0%	8.9%
	Liu and Perez [49]	79.4%	0.0%	38.6%	48.7%	1.4%	12.6%
	Manning and Eric [54]	—	—	—	48.0%	1.5%	13.2%
	Seo et al. [66]	96.0%	—	76.8%	51.1%	—	16.5%
	Williams et al. [98]	100%	100%	84.5%	55.6%	1.9%	28.4%
RL	PG	98.8%	95.6%	84.0%	53.1%	1.6%	23.4%
	PGGAN	**100%**	**100%**	**86.8%**	**65.5%**	**2.7%**	**34.6%**

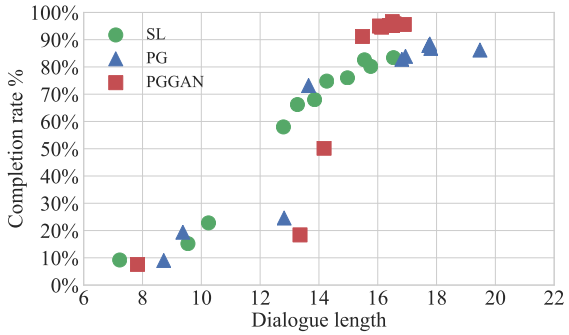


Figure 5.8: Completion rate v.s. Dialogue length on "Task5-OOV"

5.4.5 Discussion

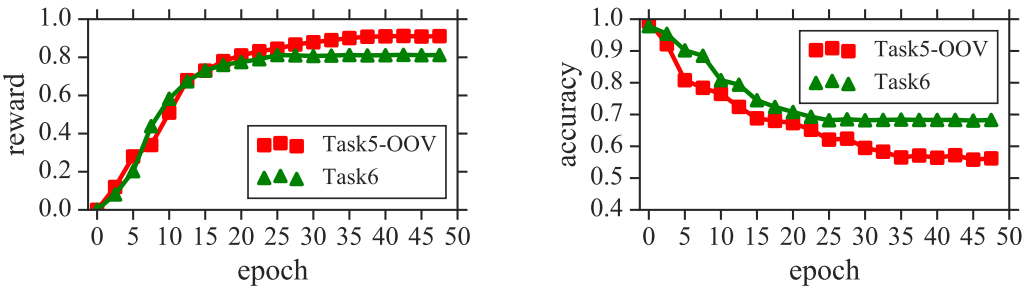
As Table 5.1 shows, when training with the same number of dialogues, the completion rate of PGGAN is much larger than the baseline models. We further illustrate the relationship between completion rate and dialogue length in Figure 5.8. The models SL, PG, and PGGAN can achieve the maximum completion rates when the dialogue length is between 15 and 18. Specifically, as shown in Table 5.2, PGGAN takes a shorter dialogue (approximately 16.5) to complete a task than PG (over 17.5) but achieves much higher completion rates, which is non-trivial to improve the user experience. Furthermore, Table 5.2 also shows that RL models (PG and PGGAN) without MC usually fail the task in less than 5 dialogue turns. It means that simply fusing policy gradient and GAN such as IRGAN [85] does not fit dialogue policy learning. Monte Carlo simulation plays a determinant role in the completion of dialogues.

We further compare PGGAN with some existing models on both "Task5-OOV" and "Task6" in turn accuracy, dialogue accuracy, and completion rate. As shown in Table 5.3, PGGAN significantly outperforms existing models in terms of all metrics on both tasks. Particularly, Table 5.4

Table 5.4: An example of the discriminator.

Dialog Context	
User Input	**KB**
USER: Hi.	N/A
AGENT: Hello, what can i help you with today?	N/A
USER: I'd like to book a table for two.	N/A
AGENT:	

Candidate Responses	**Reward**
Which price range are looking for	0.877
How many people would be in your party	0.474
You're welcome	0.103
...	...



(a) Generator rewards

(b) Discriminator accuracy

Figure 5.9: PGGAN on "Task5-OOV" and "Task6". (a) Generator rewards — the rewards that the generator obtains from the discriminator; (b) Discriminator accuracy — whether the discriminator can differentiate generated responses from human responses. The experiment results are averaged over 5 runs.

shows an example of how the discriminator evaluates candidate responses given a dialog context. It outputs a score between 0 and 1 which represents whether a generated response resembles a human response. We further plot the learning curve of the generator in Figure 5.9a and the discriminator in Figure 5.9b. The generator and the discriminator reach an equilibrium at the late training stage, meaning that the generator can respond like a real human and the discriminator cannot differentiate them.

5.5 Summary

In this chapter, we presented a task-oriented dialogue system built with the proposed adversarial dialogue policy to handle long dialogues. Through interacting with a user simulator, our dialogue system can considerably reduce the number of training dialogues. More importantly, we proposed a model PGGAN that overcomes the sparse rewards problem in dialogue policy learning via re-

inforcement learning. We evaluated the PGGAN model with the task of making restaurant reservations. The experimental results show that PGGAN significantly outperforms the state-of-the-art baselines in the completion rate (up to 25% improvement).

Chapter 6
Conclusions and Future Work

In this chapter, we summarize the main findings of this thesis in Section 6.1 and discuss future work in Section 6.2.

6.1 Conclusions

In this thesis, we focused on applying reinforcement learning techniques to policy learning for task-oriented dialogue systems. We proposed i) a *parameterized auxiliary asynchronous advantage actor-critic* (PA4C) model, and ii) a *policy gradient with generative adversarial nets* (PG-GAN) model to improve the performance of task-oriented dialogue systems.

In Chapter 4, we proposed a *context-uncertainty-aware dialogue policy* to address the problem of context uncertainties when dealing with ambiguous user inputs. In particular, when the confidence scores given by ASR and NLU components do not reach a pre-defined threshold, existing dialogue systems simply confirm with users until the confidence scores reach the threshold. However, these confidence scores vary from person to person due to different language fluency, which causes uncertainties in a dialogue context. A fixed threshold cannot adapt to these uncertainties and often leads to lengthy dialogues. We therefore proposed a context-uncertainty-aware dialogue policy that can dynamically learn a threshold according to the dialogue context. Such a dialogue policy can significantly reduce the length of dialogues and improve the user experience. We further proposed a novel reinforcement learning model named PA4C to train the context-uncertainty-aware dialogue policy. The PA4C model overcame the problem of large action spaces in existing reinforcement learning models by introducing the action parameterization technique. In particular, unlike traditional models whose output is a one-hot vector, the PA4C model can out-

put two channels to learn dialogue acts and slots separately. This approach can avoid the quadratic combination of dialogue acts and slots, and thus reduce the size of the dialogue action space. Furthermore, PA4C introduces two auxiliary tasks (reward prediction and value function replay) to help reinforcement learning models efficiently capture observed rewards. We evaluate the PA4C model on the task of calendar event creation. Experimental results show that the PA4C model significantly outperforms state-of-the-art models in task completion rate.

In Chapter 5, we proposed an *adversarial dialogue policy* to overcome the sparse reward problem in handling long dialogues. As mentioned earlier, a global reward is available only at the end of a dialogue while there are no local rewards to evaluate the quality of the response in each intermediate dialogue turn. This causes the received rewards to become sparse and cannot provide effective guidance for reinforcement learning models to explore responses. As a result, existing models may not be able to learn the optimal action. To address this issue, we proposed a novel model named PGGAN to train the adversarial dialogue policy by systematically fusing policy gradient (PG) and generative adversarial nets (GAN). PGGAN makes use of a discriminative model (i.e., discriminator) to provide local rewards for each intermediate dialogue turn, which can solve the sparse reward problem. Specifically, the discriminator is trained to differentiate the responses generated by the reinforcement learning agent (i.e., generator) from human responses, and to output a score as the reward representing the similarity between generated responses and human responses. In contrast, the generator needs to generate more human-like responses to fool the discriminator so that it can obtain larger rewards from the discriminator. Overall, the training process of the generator and the discriminator resembles a minimax game, which can be optimized alternatively. We expect the generator to win the game, meaning that the generator successfully confuses the discriminator and can respond like a real human. Furthermore, to avoid overfitting the generator to obtain large local rewards but ignoring the global reward, we integrated Monte-Carlo simulation with PGGAN. Therefore, the reward in each turn incorporates a local reward representing whether the generated response resembles a human response as well as a global reward indicating the completion or failure of user tasks, making the PGGAN model much more robust. We evaluate PGGAN with the task of making restaurant reservations. Experimental results confirm that PGGAN outperforms the state-of-the-art models in terms of task completion rates.

6.2 Future Work

Our future work will focus on the following two aspects: i) multi-domain tasks, and ii) proactive recommendation, which are discussed in Section 6.2.1 and Section 6.2.2, respectively.

6.2.1 Multi-Domain Tasks

In Chapter 4, we proposed PA4C to handle ambiguous user inputs and evaluated PA4C in the domain of creating calendar events. However, the number of dialogue acts and slots in a single domain is small (less than 10) and therefore PA4C cannot show its scalability. In our future work, we aim to apply PA4C to multiple domains (i.e., *multi-domain task-oriented dialogue systems* [92]) by increasing the number of dialogue acts and slots. Multi-domain task-oriented dialogue systems can help users complete multiple tasks within a single dialogue. They are attracting more and more attention from both research communities and the industry. For example, after users finish booking a flight, they probably would like to book a hotel. To enable such functionalities, existing studies train multiple independent local models for each domain and then aggregate these local models to a global model [18]. Such an approach does not take into account the correlation among these domains. As a result, when a new domain is introduced, they have to build an additional local model for this domain and train this model using newly collected dialogues from scratch without making use of the knowledge in existing domains. We plan to address this issue by transfer learning techniques that can transfer a part of knowledge in existing domains to the new domain, instead of training from scratch.

Recall that in Chapter 4, our proposed PA4C model separates the model output into two channels: i) a function channel to learn dialogue acts (e.g., the act "confirm"), and ii) a parameter channel to learn slots (e.g., the slot "time"). Such a structure would also suit multi-domain tasks by increasing M (i.e., dialogue acts) and N (i.e., slots). This is because many relevant domains share similar action spaces. For example, the actions "welcome", "request", and "confirm" are suitable for both flight booking and hotel booking. On the other hand, the slots of different domains are various. The available slots for flight booking may be "from city", "to city", "departure time", and "class type" while those for hotel booking may be "city", "check-in time", "days", etc. Therefore, we can retain the function channel to learn the dialogue acts across all domains but use

multiple parameter channels to learn the slots for each domain separately. When a new domain is added, we just need to add a parameter channel to the trained model and then fine-tune the newly added channel by fixing the model parameters of other domains. In this way, we can not only exploit the knowledge in existing domains, but also reduce the number of model parameters in training.

6.2.2 Proactive Recommendation

We also plan to incorporate proactive recommendation into task-oriented dialogue systems according to user contexts (e.g., time and location). Existing task-oriented dialogue systems such as Apple Siri and Microsoft Cortana are passively launched by users. When using these dialogue systems, users need to inform their intents to the systems first, and then these dialogue systems can serve the users by requesting information, which may lead to lengthy dialogues. To address this issue, we aim to build a task-oriented dialogue system that can proactively predict user intents at certain time and location. For example, when a user is leaving a shopping center at 6 pm, our dialogue system can proactively ask the user whether to request for a taxi and intelligently set her home as the destination. Therefore, the user just needs to confirm the request, which is much easier. As another example, when a user looks for a restaurant for dinner, existing task-oriented dialogue systems simply list all matching restaurants. Our future work aims to proactively recommend the best restaurants to the user according to the current dialogue context (e.g., the user's preferences) and user context (e.g., the booking time and user's location). Doing so can significantly improve user experience and encourage the use of dialogue systems.

Bibliography

[1] Rodrigo Agerri and German Rigau. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238:63–82, 2016.

[2] Charles W. Anderson, Minwoo Lee, and Daniel L. Elliott. Faster reinforcement learning after pretraining deep networks to predict state dynamics. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2015.

[3] Layla El Asri, Jing He, and Kaheer Suleman. A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Proceedings of International Speech Communication Association (INTERSPEECH)*, pages 1151–1155, 2016.

[4] Harald Aust, Martin Oerder, Frank Seide, and Volker Steinbiss. The philips automatic train timetable information system. *Speech Communication*, 17(3-4):249–262, 1995.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 30–37. 1995.

[7] Marc Bellemare, Joel Veness, and Michael Bowling. Sketch-based linear value function approximation. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2213–2221, 2012.

[8] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep

representations. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 552–560, 2013.

[9] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 226–234, 2014.

[10] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 560–564. IEEE, 1995.

[11] Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[12] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):0, 1991.

[13] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 89–96. ACM, 2005.

[14] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 129–136. ACM, 2007.

[15] Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Proceedings of International Speech Communication Association (INTERSPEECH)*, pages 3245–3249, 2016.

[16] Kenneth Mark Colby. Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(4): 515–534, 1981.

[17] Heriberto Cuayáhuitl. Simpleds: A simple deep reinforcement learning dialogue system. In *IWSDS*, pages 109–118, 2016.

[18] Heriberto Cuayahuitl, Seunghak Yu, Ashley Williamson, Jacob Carse, et al. Scaling up deep reinforcement learning for multi-domain dialogue systems. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 3339–3346, 2017.

[19] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 484–495, 2017.

[20] John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. Gemini: A natural language system for spoken-language understanding. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 54–61, 1993.

[21] Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8367–8371. IEEE, 2013.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[23] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[24] Narendra K. Gupta, Gökhan Tür, Dilek Hakkani-Tür, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. The at&t spoken language understanding system. *IEEE Trans. Audio, Speech & Language Processing*, 14(1):213–222, 2006.

[25] Hado V Hasselt. Double q-learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2613–2621, 2010.

[26] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015.

[27] Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 467–471, 2013.

[28] Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, 2014.

[29] Matthew Henderson, Blaise Thomson, and Steve J. Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of Spoken Language Technology Workshop (SLT)*, pages 360–365. IEEE, 2014.

[30] Ryuichiro Higashinaka, Mikio Nakano, and Kiyoaki Aikawa. Corpus-based discourse understanding in spoken dialogue systems. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 240–247, 2003.

[31] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[32] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of International Conference on Multimedia Information Retrieval (MIR)*, pages 39–43. ACM, 2008.

[35] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary

tasks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[36] John F Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984.

[37] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86.

[40] Sungjin Lee and Maxine Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 414–422, 2013.

[41] Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. An isu dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 119–122, 2006.

[42] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Learning dialogue strategies within the markov decision process framework. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 72–79. IEEE, 1997.

[43] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1192–1202, 2016.

[44] Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung

Chen. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016.

[45] Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

[46] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016.

[47] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[48] Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[49] Fei Liu and Julien Perez. Gated end-to-end memory networks. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1–10, 2017.

[50] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 469–477, 2016.

[51] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[52] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 285–294, 2015.

[53] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2016.

[54] Christopher D. Manning and Mihail Eric. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 468–473, 2017.

[55] Scott McGlashan, Norman Fraser, Nigel Gilbert, Eric Bilange, Paul Heisterkamp, and Nick Youd. Dialogue management for telephone information systems. In *Proceedings of the Applied Natural Language Processing Conference (ANLP)*, pages 245–246, 1992.

[56] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[57] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[58] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[59] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

[60] Tim Paek and Eric Horvitz. Conversation as action under uncertainty. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 455–464. Morgan Kaufmann Publishers Inc., 2000.

[61] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[62] Olivier Pietquin and Helen Hastie. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(1):59–73, 2013.

[63] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014.

[64] Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 438–449, 2017.

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

[66] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*, 2016.

[67] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 3776–3784, 2016.

[68] Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 3288–3294, 2017.

[69] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 3295–3301, 2017.

[70] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 3626–3633. IEEE, 2013.

[71] Bayan Abu Shawar and Eric Atwell. Chatbots: are they really useful? In *Ldv forum*, volume 22, pages 29–49, 2007.

[72] Andrew Simpson and Norman M Eraser. Black box and glass box evaluation of the sundial system. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, 1993.

[73] Florian Strub, Harm De Vries, Jeremie Mary, Bilal Piot, Aaron Courville, and Olivier Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems harm de vries. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2765–2771, 2017.

[74] Pei-hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve J. Young. Learning from real users: rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Proceedings of International Speech Communication Association (INTERSPEECH)*, pages 2007–2011, 2015.

[75] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2440–2448, 2015.

[76] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. *U. Toronto, Tech. Rep. UTML TR*, page 2010, 2010.

[77] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063, 2000.

[78] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

[79] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[80] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Natural language learning at HLT-NAACL*, pages 142–147, 2003.

[81] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 2094–2100, 2016.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 6000–6010, 2017.

[83] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.

[84] Richard S Wallace. The anatomy of alice. In *Parsing the Turing Test*, pages 181–210. Springer, 2009.

[85] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 515–524, 2017.

[86] Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 423–432, 2013.

[87] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1995–2003, 2016.

[88] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4), 1992.

[89] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

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

[91] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[92] Jason Williams. Multi-domain learning and generalization in dialog state tracking. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 433–441, 2013.

[93] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 404–413, 2013.

[94] Jason D Williams. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):959–970, 2012.

[95] Jason D Williams. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291, 2014.

[96] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[97] Jason D Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*, 2016.

[98] Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 665–677, 2017.

[99] Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building task-oriented dialogue systems for online shopping. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 4618–4626, 2017.

[100] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 2868–2876, 2017.

[101] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1321–1331, 2015.

[102] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[103] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[104] Tiancheng Zhao and Maxine Eskénazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 1–10, 2016.