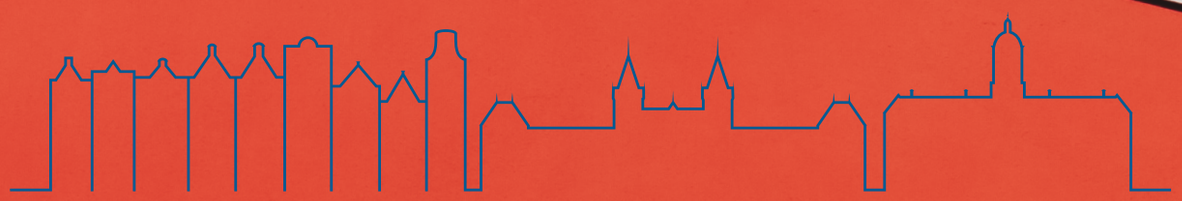
Optimizing Interactive Systems with Data-driven Objectives Ziming Li

Optimizing Interactive Systems with Data-driven Objectives

Ziming Li

HI, HOW CAN I HELP?



Optimizing Interactive Systems with Data-driven Objectives

Ziming Li

Optimizing Interactive Systems with Data-driven Objectives

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op 10 december 2020, te 13:00 uur

door

Ziming Li

geboren te Gansu

Promotiecommissie

Promotor:		
	prof. dr. M. de Rijke	Universiteit van Amsterdam
Co-promotor:		
	dr. Y. Kiseleva	Microsoft
Overige leden:		
	prof. dr. A.P.J. van den Bosch	Meertens Instituut KNAW
	prof. dr. R. Fernandez Rovira	Universiteit van Amsterdam
	dr. H.C. van Hoof	Universiteit van Amsterdam
	prof. dr. E. Kanoulas	Universiteit van Amsterdam
	dr. S. Lee	Amazon

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the China Scholarship Council.

Cover by Ziming Li
Printed by Off Page, Amsterdam, The Netherlands

ISBN: 978-94-93197-37-4

Acknowledgements

Doing a PhD brought so much more to me than what I expected four years ago. It is not merely about publishing papers or improving research skills. More importantly, this PhD journey pushed me to become more independent, confident and mature. I have grown to become a person who can think his life from more diverse perspectives and who can deal with difficulties patiently and calmly. I really appreciate my supervisors, friends, and family who have supported me to become the person I am today.

I want to thank Maarten de Rijke for being my supervisor and promotor during this PhD journey. Without your support and patience in the last four years, it would have been impossible for me to finish this book. In the first year of my PhD, I got lost in the research directions, and I was so frustrated. Thank you for being very patient and supportive. We had many discussions to clarify my research interests and what I can do in this journey. I also want to thank you for recommending the biking routes around Amsterdam and playing squash with me. They help me find the balance between doing research and enjoying life. I learned a lot from you in the last four years, I mean, not only research and squash skills but also how to be an upright person. Thank you, Maarten.

I want to thank my co-promotor, Julia Kiseleva. You introduced the initial idea of data-driven optimization and encouraged me to dig deeper in this direction. I enjoyed working with you on this research. I also appreciate what you have done to help me jump out of my comfort zone. Thank you for introducing me to different researchers during conferences, which helped me build a broader network. Besides, you are also the person who introduced squash to me, and it even became the regular sport in our office. You are not only my co-promoter but also one of my best friends in this journey.

It is my honor to have a committee consisting of very talented researchers with different backgrounds. Evangelos, Raquel, Herke, Antal, Sungjin, thank you for taking the time to read my thesis and providing valuable feedback. I also thank my paranymphs, Chang and Spyretta, for standing by me during the defense.

It was a great pleasure and honor to work with so many talented and interesting people at ILPS. I enjoyed spending time with you over coffee breaks, Friday drinks, and other social events. Thanks a lot: Aleksandr, Alexey, Ali, Ali, Amir, Ana, Anna, Antonios, Arezoo, Arianna, Artem, Bob, Boris, Chang, Chuan, Christof, Christophe, Dan, Dat, David, Dilek, Evangelos, Georgios, Gabriel, Hamid, Harrie, Hinda, Hongyu, Hosein, Ilya, Ivan, Jiahuan, Jie, Jin, Julia, Julien, Kaspar, Katya, Ke, Maarten, Maarten, Maartje, Mahsa, Mariya, Marlies, Marzieh, Maurits, Mohammad, Mostafa, Mozhdeh, Nikos, Olivier, Peilei, Petra, Pooya, Praveen, Ridho, Rolf, Sam, Sami, Sebastian, Shaojie, Spyretta, Svitlana, Tom, Vera, Wanyu, Xiaohui, Xiaojuan, Xinyi, Yangjun, Yifan, and Zhaochun. Many thanks to Xinyi and Chuan for your help in starting my life in the Netherlands. Thank you Bob for translating my thesis summary to Dutch. Thank you Petra for helping me take care of many details about my PhD program. I also want to thank all my friends from outside of ILPS. My life is getting more colorful because of you. Thank Jian and Shihan for lots of helpful discussions on both research and life. Thanks a lot: Fei, Jenny, Ji, Wei, Panos, Weina, Weiwei, Wenyan.

I want to thank my colleagues during the internships at Microsoft and Amazon.

Thank you Sungjin for being my mentor, and I appreciate all the brainstorms we had together. Many thanks to Baolin, Jinchao, and Dookun for your help and support in finishing the projects.

I want to acknowledge the China Scholarship Council (CSC) for the financial support that funded parts of the research discussed in this dissertation.

Last but not least, I would like to thank my parents for their support and encouragement. Regardless of ups and downs, you are my most powerful backing. I also want to thank my cat, Yiyue, for her accompany, and my life has been so full of laughter with her joining.

Ziming
October, 2020

Contents

1 Introduction

Interactive systems [144] play an important role in assisting people in a wide range of tasks. For instance, if users are seeking information, interactive systems can assist them in the form of web search engines [9, 19, 150, 151], dialogue systems [21, 68, 75, 100, 149], digital assistants [54, 57, 58, 135], recommender systems [116, 119], or virtual reality [3]. These systems are characterized by repeated interactions with humans that follow the request-response schema, where the user takes an action, followed by a response from the interactive system. Understanding user objectives and acting accordingly is a difficult task, even for humans [102]. What the system wants to achieve is defined by an objective function. A handcrafted objective function is heavily based on domain knowledge; it is expensive to maintain, and it does not generalize across different tasks, e.g., clicks on search results [89], gestures for mobile digital assistants [57, 151], the cross-entropy between generated replies and predefined answers [18, 68]. Consequently, manually crafted objective functions rarely correspond to the actual user experience. Currently, to design an appropriate objective function for an interactive system, a strong and comprehensive background about the domain at hand is essential, but may not be sufficient. In this dissertation, our main goal is to mine objectives directly from user interactions and use them to optimize the interactive system. As an initial step towards this goal, we propose a general framework for optimizing interactive systems and then apply it to the optimization of dialogue systems.

Dialogue systems are typical cases [126, 142] of interactive systems where the dialogue agents communicate with the user in natural language (e.g., texts, speech). According to the usage scenarios, we can group dialogue systems into two different categories: *open-domain dialogue systems*[1] and *task-oriented dialogue systems*.

Open-domain dialogue systems focus more on entertaining users [68, 105, 108, 123, 154]. Given a dialogue context, the expected return from the system should be sensible and informative to maximize the user's engagement, which is defined as the user's interest to continue the conversation in each turn [157]. There are two broad directions for training a chatbot:

1. the first strategy employs defined rules or templates to construct possible responses [138];

2. the second one builds a chatbot to learn the response generation model with a

[1] Also called chit-chat, chatbot.

machine translation framework from dialogue collections [120, 121, 123, 128].

However, in dialogue generation, the trained model may suffer from generating non-informative and generic responses such as "I don't know" [66, 69, 120, 128]. Li et al. [66] explain this behavior as "by optimizing for the likelihood of outputs given inputs, neural models assign a high probability to 'safe' responses." To address this issue, Li et al. [68] introduce a neural Reinforcement Learning (RL) generation method to generate coherent and interesting dialogues by optimizing the manually defined reward function covering ideal dialogue properties. However, the manually defined reward function is expensive to maintain and does not generalize over different domains [27, 32]. It is challenging to design a reward function for open-domain dialogue systems due to the diverse topic domains and it is not even clear what is essential to build a proper reward function when conversation scenarios are getting more and more complex. Inspired by the success of adversarial learning in computer vision, Li et al. [69] use adversarial training for dialogue generation. In [69], during the training of the generator, the reward of each generated word during decoding is estimated with the Monte Carlo search. However, the proposed method still suffers from mode collapse commonly seen in adversarial training [4]. In this thesis, we try to address the described issue of stabilizing the adversarial training process. First, we extend the adversarial dialogue generation method by incorporating an entropy regularization term to the generation objective function. Then we adopt Adversarial Inverse Reinforcement Learning (AIRL) [27, 32, 153] to train a dialogue generation model to make use of the efficient adversarial formulation and recover a more precise reward function for open-domain dialogue training. We design a specific reward function structure to measure the reward of each word in generated sentences while taking account of the dialogue context.

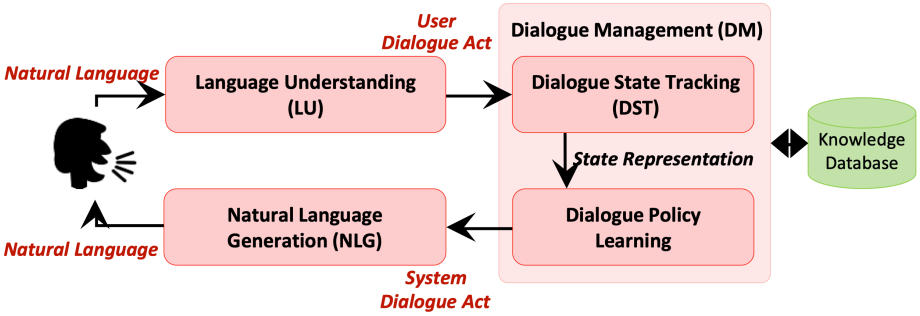


Figure 1.1: The pipeline of task-oriented dialogue systems [71].

Task-oriented dialogue systems (TDSs) aim to assist users with a certain type of task via an interactive conversational interface. Actively studied applications of TDSs range from booking movie tickets, ordering a pizza, providing traveling directions to making a phone call, or sending a message. Figure 1.1 shows the traditional solution to building such interactive systems and the pipeline can be decomposed into several sequential steps, including language understanding (LU), dialogue management (DM), natural language generation (NLG) [110, 166, 167]. One critical component of dialogue management is to decide the next action that the dialogue system should take at each

turn given the dialogue context corresponding to the history interactions. It is essential to have users in the training loop who provide feedback to make use of RL techniques in task-oriented dialogue systems. Dialogue policies can be trained and optimized automatically from scratch with user feedback [34, 130] while usable dialogue collections are not available. While interacting with real users, the dialogue agent adjusts its response policy to maximize the positive feedback given by users. The user feedback could be in the form of the user experience score at the end of the dialogue. However, it is not always practical to have real users in the training loop because it could be extremely expensive and time-consuming. Besides, frequently asking explicit feedback from users might increase the risk of degenerating the overall user experience. To overcome this problem, a popular solution is to replace real users with a user simulator along with a reward function in the training loop. The user simulator is responsible for mimicking human behavior while the reward function should return explicit feedback about the overall user experience at the end of each dialogue [70, 115]. The reward estimator embedded in the user simulator directly influences the dialogue policy from the training process to the final system performance. A commonly used reward function works as follows: when the dialogue ends, a huge positive value will be returned if the dialogue succeeds, otherwise a huge negative value as a penalty; in terms of the ongoing dialogue turns before the end step, a small negative reward is supplied for each turn to encourage shorter interactions. The problem is that dialogue agents utilizing this reward function suffer from sparse and unstable reward signals. Besides, this reward function only takes into account the task completion status and this could lead to a dialogue policy with a high success rate but poor user experience. For example, redundant system actions may not result in a failed dialogue status but it can degenerate the user experience during the interaction. Moreover, designing an appropriate and accurate reward function strongly depends on domain knowledge, and as a result, it is expensive to maintain [138]. Therefore, it is desired to learn rewards automatically directly from user interactions.

A number of adversarial learning methods have been presented recently to learn the reward function together with the dialogue policy for TDSs. Liu and Lane [83] propose to train jointly two systems:

1. a policy model that decides on the actions to take at each turn;

2. a discriminator that marks a dialogue as being successful or not.

Feedback from the discriminator is used as a reward to push the policy model to complete a task indistinguishably from humans. Improving upon this solution, Takanobu et al. [133] suggest to replace the discriminator with a reward function that acts at the dialogue action level and returns the reward for the given action relying on the dialogue state, system action, and next dialogue state as its input. However, to update the dialogue policy and the reward model on the fly in an alternating manner, we are limited to policy gradient-based algorithms, such as REINFORCE [152] and PPO [118]. Furthermore, an alternating training schema for the dialogue agent and the reward model can easily get stuck in local optima or result in mode collapse [32, 37, 42, 156]. Targeting this problem, we propose a new approach for training a dialogue policy by decomposing the adversarial learning method into two sequential steps. First, we learn the reward function using an auxiliary dialogue state generator where the loss from the discriminator can

be backpropagated to the generator directly. Second, the trained discriminator as the dialogue reward model will be incorporated into the RL process to guide dialogue policy learning and will not be updated, while the state generator is discarded. Therefore, we can utilize any RL algorithm to update the dialogue policy, including both on-policy and off-policy methods.

As mentioned above, to train a dialogue policy with reinforcement learning, we should have a user simulator and a reliable reward function. Currently, most user simulators are rule-based and they have the same disadvantages (e.g., hard to maintain, domain knowledge required, issues with scalability) as rule-based dialogue systems [138]. The difference is that rule-based approaches to system design meet this problem at the dialogue agent side while rule-based user simulators need to solve it at the environment side. Furthermore, a set of labeled human-human dialogues are essential if we want to apply adversarial learning to learn a reward function. Building realistic user simulators and collecting high-quality labeled dialogues are making it more and more challenging to train a high-quality dialogue agent with RL [70, 133]. It is meaningful if we can rethink RL methods in dialogue policy learning and revisit some traditional learning methods for task-oriented dialogue systems. In this thesis, we pick up supervised learning and exploit its potential for dialogue policy training. Supervised learning does not suffer from the need to design and maintain a complex user simulator. Although it requires labeled dialogues, collecting labeled data is becoming more feasible and more importantly, we can reuse dialogues from adversarial dialogue policy learning methods [83, 133] without having to collect new resources.

1.1 Research Outline and Questions

In this dissertation, our main goal is *to learn objectives directly from user interactions and use them to optimize an interactive system*. To this extend, we propose a general framework for optimizing interactive systems with data-driven objectives (Chapter 2) and explore the possibility of applying the proposed framework to real applications, including open-domain dialogue systems (Chapter 3), and task-oriented dialogue systems (Chapter 4 and 5). Below we introduce the main research questions studied and answered in each chapter.

1.1.1 Optimizing interactive systems with data-driven objectives

Previous work on interactive systems has relied on the assumption that handcrafted objective functions can accurately reflect users' preferences and intentions while interacting with interactive systems. As a result, interactive systems have been optimized for manually designed objectives that do not always align with the true user preferences which lead to the inability to generalize across different domains. This brings us to the first research question:

RQ1 Can interactive systems be optimized using objectives recovered from user interactions directly?

To answer this question, we propose a novel two-step framework in Chapter 2 to optimize interactive systems with data-driven objectives. We first infer a user reward model given collected user interaction traces and then update the system with the inferred reward functions via a novel algorithm: the Interactive System Optimizer (ISO). We model user-system interactions using Markov Decision Process (MDP), where the agent is the user and the stochastic environment is the interactive system. Then we formalize an optimization problem to infer user needs from observed user-system interactions, in the form of a data-driven objective. Importantly, our method works without any domain knowledge and is thus even applicable when prior knowledge is absent. Eventually, we propose a novel, Interactive System Optimizer, that alternates between optimizing the interactive system for the current inferred objective and letting the user adapt to the new system behavior. To validate the success of the proposed method, we apply it to two different simulated interactive systems and show that ISO robustly improves the system performance.

1.1.2 Optimizing open-domain dialogue systems with data-driven objectives

Dialogue systems are typical cases of interactive systems where the dialogue agents communicate with the user in natural language. RL methods have emerged as a popular choice for training an efficient and effective dialogue policy. However, these methods suffer from sparse and unstable reward signals returned by a user simulator only when a dialogue finishes. Besides, the reward signal is manually designed by human experts, which requires domain knowledge. Recently, a number of adversarial learning methods have been proposed to learn the reward function together with the dialogue policy. With respect to the open-domain dialogue system, the attempt of using adversarial training [69] for dialogue generation still suffers from mode collapse which is commonly seen in adversarial training. This brings us to our second research question:

RQ2 Can data-driven reward functions be used to successfully improve open-domain dialogue systems?

To answer RQ2, we first extend a recently proposed adversarial dialogue generation method [69] to an adversarial imitation learning solution by incorporating an entropy regularization term to stabilize the adversarial training process. Then, in the framework of adversarial inverse reinforcement learning, we propose a new reward model for dialogue generation that can provide a more accurate and precise reward signal for generator training. We evaluate the performance of the resulting model with automatic metrics and human evaluations and demonstrate that our model can generate more high-quality responses and achieve higher overall performance.

1.1.3 Optimizing task-oriented dialogue systems with data-driven objectives

The success of adversarial training for dialogue generation also leads to attempts to optimize task-oriented dialogue systems with a self-learned reward function [83, 133].

These methods propose to learn dialogue rewards directly from dialogue samples, where a dialogue agent and a dialogue discriminator are trained in an alternating manner. However, this training schema is only applicable to policy gradient-based algorithms and off-policy algorithms are missing (e.g., DQN [91]) in adversarial dialogue policy learning. Therefore, we explore if off-policy based methods can also benefit from the high-quality reward function from adversarial training. This brings us to the third research question:

RQ3 Can off-policy RL methods benefit from data-driven objectives in dialogue policy learning for TDSs?

To answer RQ3, we propose to decompose adversarial training into two consecutive steps. First, we train the discriminator with an auxiliary dialogue generator and then incorporate a derived reward model into a common RL method to guide the dialogue policy learning. This approach is applicable to both on-policy and off-policy RL methods. Based on our extensive experimentation, we can conclude that the proposed method: (1) achieves a remarkable task success rate using both on-policy and off-policy RL methods; and (2) has the potential to transfer knowledge from existing domains to a new domain.

RL and Inverse Reinforcement Learning (IRL) have brought great progress to dialogue policy learning for TDSs according to the current evaluation mechanisms. However, these methods are also becoming more and more sophisticated [70, 83, 101, 133]. More uncontrollable factors are involved (e.g., mode collapse [83]) and the requirements to deploy such methods are also becoming more strict (e.g., high-quality user simulators [70], labeled dialogues for adversarial training [83, 133]). The described issues suggest us to reconsider the "pure" use of RL methods in TDSs[2] and revisit some traditional dialogue training methods. That observation brings us to the following research question:

RQ4 Are we really making progress in applying only RL to dialogue policy learning for TDSs?

We demonstrate how (1) traditional supervised learning together with (2) a simulator-free adversarial learning method can be used to achieve performance comparable to state-of-the-art (SOTA) RL based methods. With respect to the supervised learning methods, we first introduce a simple dialogue action decoder to predict the appropriate actions. Then, the traditional multi-label classification solution for dialogue policy learning is extended by adding dense layers to improve the dialogue agent performance. In terms of the simulator-free adversarial learning method, we employ the Gumbel-Softmax estimator to train the dialogue agent and the dialogue reward model without using RL. Based on our extensive experimentation, we conclude that the proposed methods can achieve more stable and higher performance with fewer efforts, such as the domain knowledge required to design a user simulator and the intractable parameter tuning in reinforcement learning. According to the human evaluation results, the simple dialogue action decoder can bring the highest user experience ratings.

[2]It is reasonable to assume that adversarial training in open-domain dialogue systems may face the same problems but in this work, we only focus on TDSs.

1.2 Main Contributions

In this section, we list the theoretical, algorithmic, and empirical contributions of this thesis. For each contribution, we list the chapter from which it originates.

1.2.1 Theoretical contributions

1. A novel algorithm, Interactive System Optimizer (ISO), that optimizes an interactive system through data-driven objectives (Chapter 2).

1.2.2 Algorithmic contributions

2. A new way of modeling user-system interactions, where a user is the agent while a system is the environment (Chapter 2).

3. A novel optimization setup to infer data-driven objectives that accurately reflect the users' needs solely from interactions, without using any domain knowledge to handcraft an optimizing goal (Chapter 2).

4. A novel word-level reward model architecture to evaluate the reward of each word in a dialogue, which enables us to have a more accurate signal compared to existing turn-level reward architecture for adversarial dialogue training (Chapter 3).

5. A novel Seq2Seq model, Dialogue Generation with Adversarial Inverse Reinforcement Learning (DG-AIRL), for addressing the task of dialogue generation built on adversarial inverse reinforcement learning (Chapter 3).

6. A reward learning method that is applicable to off-policy RL methods in dialogue training for TDSs (Chapter 4).

7. A reward learning method that can alleviate the problem of local optima for adversarial dialogue training for TDSs (Chapter 4).

8. A reward function that can transfer knowledge learned in existing domains to a new dialogue domain for TDSs (Chapter 4).

9. A dialogue action decoder to learn the dialogue policy with supervised learning for TDSs (Chapter 5).

10. A multi-label classification solution to learn the dialogue policy for TDSs (Chapter 5).

11. A simulation-free adversarial learning method to improve the performance of dialogue agents for TDSs (Chapter 5).

1.2.3 Empirical contributions

12. Two different simulated interactive systems to validate the success of the interactive system optimizer. We show how the proposed optimizer can improve the system performance in the designed setups. We also show that by inferring user reward functions, we can optimize the interactive system without real users in the loop and real users are only involved while collecting user-system interaction trajectories (Chapter 2).

13. An improvement of the training stability of adversarial training by employing causal entropy regularization for open-domain dialogue systems (Chapter 3).

14. Achieving SOTA performance in dialogue Policy Learning (PL) with fewer efforts and costs compared to existing RL based solutions for TDSs (Chapter 5).

1.3 Thesis Overview

In this thesis, we focus on exploring how to *mine objectives directly from user interactions and use them to optimize the system*.

In Chapter 1, we provide the reader with an introduction and background knowledge about the theme, and we list the main research questions of this dissertation.

In Chapter 2, as a foundation for the thesis, we propose a general framework for optimizing interactive systems without using any domain knowledge to handcraft an optimization goal. The data-driven objectives are given in the format of recovered reward functions. We verify the success of the proposed solution with two simulated experimental setups.

In Chapter 3, we apply the main idea from Chapter 2 about optimizing interactive systems with data-driven objectives, to open-domain dialogue systems. This is our first attempt of validating the effectiveness of data-driven objectives with real applications.

In Chapter 4, we extend the usage range of data-driven objectives from on-policy to off-policy RL methods for Task-oriented dialogue systems (TDSs). Next, we demonstrate the potential of making use of the recovered reward functions to transfer knowledge among different domains.

In Chapter 5, we study the downside of optimizing interactive systems with only RL and IRL in terms of TDSs. We show that it is meaningful to rethink the role of RL in training dialogue policies. Besides, it is valuable to revisit some traditional supervised learning methods to release their potential for dialogue policy learning.

Chapter 6 concludes this thesis and proposes directions for future work.

All chapters in thesis share the same goal: *mining objectives directly from user interactions and use them to optimize the system*. Chapter 2 introduces the general optimization framework and Chapters 3, 4, and 5 are our attempts at utilizing the framework for real applications, including open-domain dialogue systems and TDSs. If time is of the essence, readers can first read Chapter 2 and then pick up one chapter from Chapter 3, 4, 5 according to their interests.

1.4 Origins

In this section, we list the publications each chapter is based on and explain the role of each author.

Chapter 2 is based on the following papers:

- Z. Li, J. Kiseleva, A. Agarwal, and M. de Rijke. Learning data-driven objectives to optimize interactive systems. In *NeurIPS LIRE 2019 Workshop: Learning with Rich Experience: Integration of Learning Paradigms*, 2019.
- Z. Li, J. Kiseleva, A. Agarwal, M. de Rijke, and R. W. White. Optimizing interactive systems via data-driven objectives. *Journal of Artificial Intelligence Research*, 2020. Submitted.

The journal paper is an extension of the workshop paper. ZL, JK and MdR formulated the main research questions. AA helped with reformulating the idea and contributed to the design of the experimental setup. ZL performed the experiments and results analysis. RW joined the work for the journal version and helped with reorganizing the storyline. All authors contributed to the writing.

Chapter 3 is based on the following paper:

- Z. Li, J. Kiseleva, and M. de Rijke. Dialogue generation: From imitation learning to inverse reinforcement learning. In *AAAI*, volume 33, pages 6722–6729, 2019.

ZL designed the model and performed the experiments. JK and MdR contributed to the formulation of the idea. ZL did most of the writing.

Chapter 4 is based on the following paper:

- Z. Li, S. Lee, B. Peng, J. Kiseleva, M. de Rijke, J. Li, S. Shayandeh, and J. Gao. Guided dialogue policy learning without adversarial learning in the loop. In *EMNLP findings*, 2020.

This research was mainly performed during a research internship at Microsoft Research. ZL designed the model and finished most of the writing. SL, BP, JL, SS, and JG contributed to the initial idea during the internship. JK and MdR helped with reformulating the idea and polishing the paper.

Chapter 5 is based on the following paper:

- Z. Li, J. Kiseleva, and M. de Rijke. Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. In *EMNLP findings*, 2020.

ZL designed the model and performed the experiments. JK and MdR contributed to the formulation of the idea. ZL did most of the writing.

Work on the thesis also benefitted from work on the following publications:

- Z. Li, J. Kiseleva, M. de Rijke, and A. Grotov. Towards learning reward functions from user interactions. In *ICTIR*, pages 289–292, 2017.
- Z. Li and M. de Rijke. The impact of linkage methods in hierarchical clustering for active learning to rank. In *SIGIR*, pages 941–944, 2017.
- Z. Li, J. Kiseleva, and M. de Rijke. Generating coherent and informative responses with backward-reasoning in open-domain dialogue systems. In *NAACL*, 2021. Submitted.
- Z. Li, D. Park, J. Kiseleva, and S. Lee. Estimating user satisfaction level for multi-turn dialogues. In *NAACL*, 2021. Submitted.

2

Optimizing Interactive Systems via Data-driven Objectives

This chapter is meant to answer the research question:

> *Can interactive systems be optimized using objectives recovered from user interactions directly?*

We present a novel two-step framework to optimize interactive systems with data-driven objectives.

2.1 Introduction

Interactive systems [144] play an important role in assisting people in a wide range of tasks. For instance, if users are seeking information, interactive systems can assist them in the form of web search engines [9, 19, 150, 151], dialogue systems [21, 68, 75, 100, 149], digital assistants [54, 57, 58, 135], recommender systems [116, 119], or virtual reality [3]. The described instances of interactive systems can be considered as examples of machine learning applications where the goal is to assist users in real-world day-to-day tasks. These systems are characterized by repeated interactions with humans that follow the request-response schema, where the user takes an action, followed by a response from the interactive system. Such interactions can continue for several iterations until the user decides to stop, e.g., when they are either satisfied or frustrated with their experience. Interaction with the system produces traces/trajectories of user interactions. Importantly we assume that an interactive system and its users always have a shared goal: for users to have the best experience in the premise of completing the user's task.

Thus, both a system and its users are expected to behave accordingly, e.g., a searcher issues a query that he expects will lead him to the desired results and the interactive system provides the search results that are most helpful to him. However, despite their shared goal, only the user can observe their own experience, leaving interactive systems unable to directly optimize their behavior. For example, in most cases, users will not leave explicit rates about their experience after interacting with digital assistants and this brings difficulties in optimizing the system.

This chapter was published as [76].

Understanding user objectives and acting accordingly is a difficult task, even for humans [102]. However, studies in behavioral economics provide supporting evidence that users intend to maximize expected utility or minimize expected cost and effort [8, 17, 86, 136]. Following this line of research, in this chapter, we assume that the general population of users has a shared goal that is achieved through interactions, so user behavior is aligned with their preferences but rational noises are allowed. We call such behavior *approximately rational*. Obviously, the exact user utility function is inherently complex, but we can approximate it via some meaningful *objective function*, recovered directly from observed traces of user interactions. A similar principle has been successfully employed in robotics [36, 48, 107] and for understanding user behavior on the web [5, 59, 141]. Hence, knowing the *approximate* user objective function can help us to improve the flow of interactive systems.

Currently, optimizing interactive systems relies on explicit assumptions about users' objectives in terms of their needs and frustrations [73]. Commonly, an objective function is manually designed for a particular task to reflect the quality of an interactive system, e.g., in terms of user satisfaction [50, 51], user effort [155] or other domain-specific metrics, such as relevance judgements in information retrieval [22, 24, 47, 113, 114], or user feedback (e.g., click, order, skip) in recommender systems [14, 159, 160]. The drawbacks of this approach are that a handcrafted objective function is heavily based on domain knowledge, that it is expensive to maintain, and that it does not generalize over different tasks, e.g., clicks on search results [89], gestures for mobile digital assistants [57, 151], the cross-entropy between generated replies and predefined answers [18, 68]. Consequently, manually crafted objective functions rarely correspond to the actual user experience. Therefore, even an interactive system that maximizes an objective function is not expected to provide an optimal experience as long as that objective function is hand-crafted. Moreover, it is impossible to design such functions when there is a lack of domain knowledge. Also, we have witnessed how badly designed objective functions can lead to wrong results. For example, Liu et al. [85] validated that applying evaluation metrics (e.g., BLEU score [98]) in the machine-translation field to dialogue systems is problematic because there is significant diversity in the space of valid responses to a given context.

Given an objective function, optimization can be done following the Reinforcement Learning (RL) paradigm [132], which has been successfully applied to physically constrained environments [28, 64, 65, 127]. The majority of previous work in the area of interactive systems does this by considering the interactive system as the agent and the underlying stochastic environment induced by the user [44, 63, 68, 82, 99, 131], where the system policies are optimized by interacting with real users or user simulators. However, this setup does not allow us to apply the principle outlined earlier, that it is the user (not the interactive system) who is being rewarded by interacting with the system while maximizing their utility. Recently, Leike et al. [62] have shown how agent alignment, cast in an RL framework, can be applied for optimizing general-purpose interactive systems via reward modeling. Jeon et al. [48] and Reddy et al. [107] demonstrate how this approach can be applied in the robotics domain. However, this setup requires a quantity of user feedback that may not always be available in practice [31, 49], which leaves us with unlabeled user trajectories.

In this chapter, we assume that users continue their interactions with the system if

their goals are fulfilled, or at least partially fulfilled, so they are getting rewards after each action. we propose a general perspective on how to improve interactive systems by simultaneously (1) inferring an objective function directly from data, namely unlabeled trajectories of user interactions with the system, and (2) iteratively, and step by step, optimizing the system for this data-driven objective. Since users have difficulties in comprehending dramatic changes in an interactive system [90, 96, 134, 145], changes should be made gradually to let users adapt to a newly optimized interactive system. The proposed setup is schematically outlined in Figure 2.1. It embodies a principled approach by concurrently inferring data-driven objectives from user interactions and optimizing the interactive system accordingly. Thus, our approach does not depend on any domain knowledge.

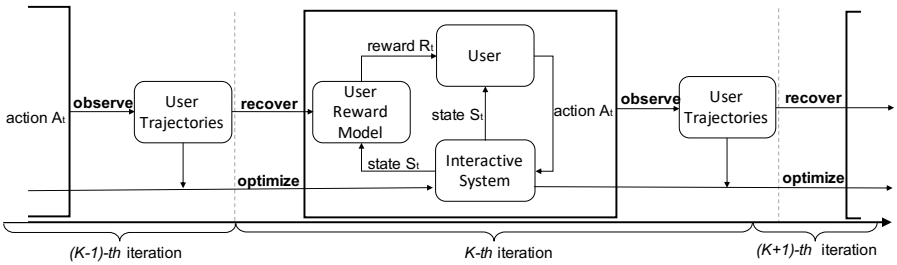


Figure 2.1: Schematic illustration of the proposed setup of iterative gradual optimization of the flow of an interactive system: the user reward model is *recovered* from the logs collected while users are interacting with the interactive system; the Interactive System Optimizer (ISO) is used to *optimize* the interactive system at each iteration.

Below, we start by outlining relevant research areas (Section 2.2). Then we make the following contributions:

- A new way of modeling user-system interactions, which is depicted as the k-th iteration in Figure 2.1 (Section 2.4).
- A novel optimization setup to infer data-driven objectives that accurately reflect the users' needs solely from interactions, without using any domain knowledge to handcraft an optimizing goal, which is partially reflected by the arrows labeled "recover" in Figure 2.1 (Section 2.5).
- A novel algorithm, ISO, that optimizes an interactive system through data-driven objectives, which is depicted with the arrow labeled "optimize" in Figure 2.1 (Section 2.6).
- Applications of ISO to two simulated interactive systems to assess the method. We show how the proposed optimizer can improve the system performance in the designed setups. We also show that by inferring user reward functions, we can optimize the interactive system without real users in the loop and real users are only involved while collecting user-system interaction trajectories (Section 2.7).

2.2 Related Work

Relevant work for this chapter comes in two broad strands: how to optimize interactive systems (Section 2.2.1) and what reward signal can be used for optimization (Section 2.2.2).

2.2.1 Optimizing interactive systems

The flow of interactive systems [144] can be improved by direct and indirect optimization. Direct optimization aims to maximize user satisfaction directly; in contrast, indirect optimization solves a related problem while hoping that its solution also maximizes user satisfaction [19]. Direct optimization can be performed using supervised learning or RL [93]. In RL, an agent learns to alter its behavior through trial-and-error interactions with its environment [132]. The goal of the agent is to learn a policy that maximizes the expected return. RL algorithms have successfully been applied to areas ranging from traditional games to robotics [2, 23, 39, 40, 64, 65, 92, 117, 118, 127, 137, 139, 161].

Many applications of RL to optimizing interactive systems come from such fields as Information Retrieval (IR), recommender systems, and dialogue systems. The general assumption of users trying to maximize their utility proposed in [48, 107] holds for interactive systems as well [5]. Hofmann et al. [43, 45] apply RL to optimize IR systems; they use RL for online learning to rank and use interleaving to infer user preferences [44]. Luo et al. [88, 89] define a reward function as the number of satisfied clicks in the search session. Shani et al. [124] describe an early MDP-based recommender system and report on its live deployment. Levin et al. [63] formulate the problem of dialogue design as an optimization problem with an objective function reflecting different dialogue dimensions relevant for a given application and they map a dialogue system to a stochastic model known as Markov Decision Process (MDP). Li et al. [68] apply RL to optimize dialogue systems; in particular, they optimize handcrafted reward signals such as ease of answering, information flow, and semantic coherence. A number of RL methods, including Q-learning [71, 82, 99, 131] and policy gradient methods [21, 133, 149], have been applied to optimize dialogue policies by interacting with real users or user simulators. With the help of RL, the dialogue agent can explore contexts that may not exist in previously observed data. A key component in RL is the quality of the reward signal used to update the agent policy. Most existing RL-based methods require access to a reward signal from user feedbacks or a predefined reward.

However it remains non-trivial to apply the RL paradigm to scalable real-world machine learning tasks [62] due to the lack of a general approach of recovering data-driven objectives, which we discuss next.

2.2.2 Rewards for interactive systems

When applying RL to the problem of optimizing interactive systems, we need to have rewards for at least some state-action pairs. Previous work typically handcrafts those, using, e.g., normalized discounted cumulative gain (nDCG) [97] or clicks [60, 159] before the optimization or the evaluation of the algorithm. Instead of handcrafting

rewards, we recover them from observed interactions between the user and the interactive system using Inverse Reinforcement Learning (IRL). The main motivation behind IRL is that designing an appropriate reward function for most RL problems is non-trivial; this includes animal and human behavior [1], where the reward function is generally assumed to be fixed and can only be ascertained through empirical investigation. Thus inferring the reward function from historical behavior generated by an agent's policy can be an effective approach. Another motivation comes from imitation learning, where the aim is to teach an agent to behave like an *expert* agent. Instead of directly learning the agent's policy, other work first recovers the expert's reward function and then uses it to generate a policy that maximizes the expected accrued reward [95]. Since the inception of IRL [112], several IRL algorithms have been proposed, including maximum margin approaches [1, 106], and probabilistic approaches [10, 81, 164]. In recent years, a number of adversarial IRL methods have been proposed because of its ability to adapt training samples to improve learning efficiency [26, 27, 32, 41, 104, 122]. Another aspect in which IRL methods differ is the availability of feedback or scores for user traces. Christiano et al. [16], Leike et al. [62] suggest a setup where the system can learn from user feedback, which is not always available in practice. In this chapter, we tackle the case without explicit feedback.

Regarding applications of IRL, Ziebart et al. [162] use IRL for predicting the desired target of a partial pointing motion in graphical user interfaces. Monfort et al. [94] use IRL to predict human motion when interacting with the environment. IRL has also been applied to dialogues to extract the reward function and model the user [75, 78, 103, 133]. IRL is used to model user behavior to make predictions about it. But we use IRL as a way to recover the rewards from user behavior instead of handcrafting them and optimize an interactive system using these recovered rewards. Lowe et al. [87] learn a function to evaluate dialogue responses. However, the authors stop at evaluation and do not optimize the interactive system.

Recent work [48, 62, 158] demonstrates impressive results and outlines a new research direction while modeling user-system interactions using the agent alignment problem [132]. In contrast to our work, reward modeling heavily relies on an explicit user feedback loop, which is rarely available in web-based interactive systems [55, 56]. The key difference between our work and previous studies is that we first use recovered rewards from observed user interactions to reflect user needs and define interactive system objectives. Subsequently, the interactive system can be optimized according to the defined data-driven objectives to improve the user experience. We regard the interactions between a user and an interactive system as an agent interacting with a changeable environment, where the transition distribution of the environment can be updated. Treating an interactive system as a changeable and programmable environment is novel and reasonable because we have complete control over the behavior of interactive systems since we are the system designers. Lowe et al. [87] learn a function to evaluate dialogue responses but do not optimize the interactive system. Leike et al. [62] formulate the optimization problem in a complex multi-agent setup because their environment is physical and non-programmable; besides, the reward modeling by Leike et al. [62] heavily relies on the user feedback loop, which is mostly not available in the interactive systems.

2.3 Background

Reinforcement Learning (RL) and Inverse Reinforcement Learning (IRL) are the fundamental techniques used in the framework we propose in this chapter.

In RL an agent learns to alter its behavior through trial-and-error interactions with its environment [132]. The goal of the agent is to learn a policy that maximizes the expected return. RL algorithms have successfully been applied to areas ranging from traditional games to robotics [2, 23, 39, 40, 64, 65, 92, 117, 118, 127, 137, 139, 161].

The task of IRL is to extract a reward function given observed, optimal (or suboptimal) behavior of an agent over time [95]. The main motivation behind IRL is that designing an appropriate reward function for most RL problems is non-trivial; this includes animal and human behavior [1], where the reward function is generally assumed to be fixed and can only be ascertained through empirical investigation. Thus inferring the reward function from historical behavior generated by an agent's policy can be an effective approach. Another motivation comes from imitation learning, where the aim is to teach an agent to behave like an *expert* agent. Instead of directly learning the agent's policy, other work first recovers the expert's reward function and then uses it to generate a policy that maximizes the expected accrued reward [95]. Since the inception of IRL [112], several IRL algorithms have been proposed, including maximum margin approaches [1, 106], and probabilistic approaches [10, 164]. In the last few years, a number of adversarial IRL methods [26, 27, 32, 41, 104, 122] have been proposed because of their ability to adapt training samples to improve learning efficiency.

2.4 Modeling User-System Interactions

In this section, we first introduce our assumptions about collaborations between a user and an interactive system (Section 2.4.1), and then we explain how we model these interactions (Section 2.4.2).

2.4.1 Assumptions

Our goal is to design an interactive system that can successfully assist users in completing some real-world tasks. We have formulated a set of assumptions to formalize user-system interactions, which are schematically depicted in Figure 2.1. They can be separated into two groups: assumptions about the system design (S) and assumptions regarding the user goals and behavior (U):

Assumption 1 **S:** the system's goal is to accommodate a better user experience, namely to maximize the utility a user gets from the system by minimizing their efforts;

Assumption 2 **S:** the system setup allows us to iteratively and gradually improve the system in a sequential manner to accommodate a better user experience, given that at the start the system provides "non-zero" utility for users but can be significantly improved;

Assumption 3 **S:** a system designer can transform an interactive system, but it has some required steps a user needs to take to complete their task due to system design constraints;

Assumption 4 **U:** users have incentives to continue their interactions with an interactive system if they are getting some value from it;

Assumption 5 **U:** users of an interactive system have *approximately* homogeneous behavior, namely users have a shared notion of utility that can be approximated by some objective function;[1]

Assumption 6 **U:** users try to maximize their utility while interacting with a system;

Assumption 7 **U:** users are not required to provide explicit feedback about their experience. However, user actions can be considered as *implicit signals* reflecting their satisfaction/frustration with an interactive system.

2.4.2 Modeling interactions

While employing a RL formalism as Assumption 5 and Assumption 6 can be reformulated as follows: the user is seen the *optimal* agent who interacts with the environment, an interactive system, with the goal of maximizing their expected rewards. As a *running example* we can consider a user who is interacting with a search engine. The process of user-system interaction is modeled using a finite MDP (S, A, T, r, γ), in the following way:[2]

- S is a set of states that represent responses from the interactive system to the user. S is finite as there is a limited predefined number of responses that the interactive system can return.
- A is a finite set of actions that the user can perform on the system to move between states. In the case of a search engine, a user can run a query, click on the returned results, reformulate a query, etc.
- T is a transition distribution and $T(s, a, s')$ is the probability of transitioning from state s to state s' under action a at time t:

$$T(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a). \tag{2.1}$$

  For search engines, being at the start page (which is s) a user is taking an action a (e.g., running a query), and the engine redirects him to a result page (which is s').

- $r(s, a, s')$ is the expected immediate reward after transitioning from s to s' by taking action a. We compute the expected rewards for (state, action, next state) triples as:

$$r(s, a, s') = \mathbb{E}[R_t \mid S_t = s, A_t = a, S_{t+1} = s'], \tag{2.2}$$

[1] The terms "users" and "a user" are used interchangeably.
[2] We follow the notation proposed in [132].

where R_t is the reward at time t. In the case of a search engine, a user is getting a reward for finding the desired information. However, the rewards are not observed in practise (Assumption 7). For simplicity in exposition, we write rewards as $r(s)$ instead of $r(s, a, s')$ in our setting; the extension is trivial [95].

- $\gamma \in [0, 1]$ is a discount factor.

We write $\mathcal{P}$ to denote the set of interactive systems, i.e., triples of the form (S, A, T). Following Assumption 3, system designers have control over the sets S, A, and the transition distribution, T, and T can be changed to optimize an interactive system.

The *user behavior strategy* for accomplishing their tasks is represented by a policy, which is a mapping, $\pi \in \Pi$, from states, $s \in S$, and actions, $a \in A$, to $\pi(a|s)$, which is the probability of performing action $A_t = a$ by the user when in state $S_t = s$. The observed history of interactions between the user and the interactive system, H,[3] is represented as a set of trajectories, $\{\zeta_i\}_{i=1}^n$, drawn from a distribution Z, which is brought about by T, π, and D_0, where D_0 is the initial distribution of states. Following Assumption 5, which proposes homogeneity in user behavior, simplifies the problem, i.e., as if a single user-generated H. A *trajectory* is a sequence of state-action pairs, where a user does not provide explicit feedback (Assumption 7):

$$\zeta_i = S_0, A_0, S_1, A_1, \ldots, S_t, A_t, \ldots. \tag{2.3}$$

To conclude, we suppose that the user is an optimal agent who is trying to maximize its reward under the system dynamics, it faces and that the system wants to improve the user experience over time by creating progressively easier MDPs to solve for the user. However, an interactive system cannot transition from all initial to goal states in one step due to design constraints. For example, if a user is searching for holiday destinations, the system cannot redirect him to the final stage of booking a hotel because he needs to go through a necessary step, e.g., providing payment details.

To summarize, we have described the basic principles of modeling interactions between users and an interactive system. Next, we detail how to define data-driven objectives that are used to optimize an interactive system.

2.5 Defining Data-driven Objectives

In this section, we first present our approach to convert user needs to data-driven objectives of an interactive system (Section 2.5.1), and then we explain how these objectives can be estimated (Section 2.5.2).

2.5.1 Defining interactive system objectives

We define the *quality* of an interactive system as the expected state value under an optimal user policy. The value of a state S_0 under a policy π is given as [132]:

$$V^\pi(S_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right], \tag{2.4}$$

[3]Below, we sometimes refer to the history H as logs of user interactions, or user trajectories/traces, or simply logs.

where the expectation $\mathbb{E}_\pi[\cdot]$ is taken with respect to sequences of states $S_0, S_1, \ldots, S_t, \ldots$ drawn from the policy π and transition distribution T. We use V_T^π to denote the value of policy π under the current transition distribution T, and hide the initial states S_0 for simplification.

In the proposed setting, the user's goal is to find the best policy π^* such that V_T^π is maximized. $V_*(T)$ defines the maximum possible value of V_T^π under transition distribution T as follows:

$$V_*(T) = \max_{\pi \in \Pi} V_T^\pi, \tag{2.5}$$

where Π is the set of possible user policies. We formulate the problem of finding the optimal interactive system's transition distribution, denoted T^*, in the following terms:

$$T^* = \arg\max_{T \in T} V_*(T). \tag{2.6}$$

Therefore, Eq. 2.6 represents the objective function, mentioned in Assumption 6, which is derived from user trajectories (Eq. 2.3) directly. After finding T^* by solving the proposed optimization problem, the system designer can transform the current system to a new one, which should deliver a better user experience as it reflects user needs better. This process is illustrated in Figure 2.1 by the arrow marked *optimize* between two consecutive iterations.

With the transition distribution T, the interactive system will respond with the next state s' given the current state s and the user action a. In real life, it is not guaranteed that the tuple (s, a, s') exists. For example, in task-oriented dialogue systems, the system first needs to collect essential information for booking a hotel (e.g., hotel name, room type) step by step. In some cases, the system also needs to recommend potential hotels and asks the user to make a choice. After successfully collecting all information, the system can guide the user to a payment page. Obviously, it is not possible to deliver a payment state to the user when the information contained in the current state is not complete. Therefore, inherent constraints exist in interactive systems and this makes finding the optimal interactive system's transition distribution a meaningful and interesting task. Otherwise, the system can always deliver the most valuable state to the user in one step at any state.

To estimate the data-driven objectives interactive system presented in Eq. 2.6, we first need to recover R_t, which we will discuss next.

2.5.2 Recovering user rewards

Assumption 4 suggests that continued user interactions with the system indicate a certain level of user satisfaction, which can be reflected by experienced rewards. In contrast with $\zeta_i \in H$ presented in Eq. 2.3, the complete history of interactions, $\hat{H}$, consists of trajectories $\hat{\zeta}_i \sim \hat{Z}$, which include the user reward R_t:

$$\hat{\zeta}_i = S_0, A_0, R_1, S_1, A_1, R_2 \ldots, R_t, S_t, A_t, \ldots. \tag{2.7}$$

The problem is that the true user reward function is hidden from an interactive system and inherently difficult due to the complexity of the real world surrounding users. Our goal is to use the collected incomplete user trajectories, H, shown in Eq. 2.3, to find

a way to approximate true user rewards. To address this challenge we apply Inverse Reinforcement Learning (IRL) methods,[4] which are proposed to recover the rewards of different states, $r(s)$, for $\zeta_i \in H$. Our assumption about the form of user reward function is: given state feature functions $\phi : S_t \rightarrow \mathbb{R}^k$ that describe S_t as a k-dimensional feature vector, the true reward function $r(s)$ is a linear combination of the state features $\phi(s)$, which can be given as $r(s) = \theta^T \phi(s)$. To uncover the reward weights θ, we employ the following approaches.

Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) The core idea of MaxEnt-IRL [164] is that trajectories with equivalent rewards have equal probability to be selected and trajectories with higher rewards are exponentially more preferred, which can be formulated as:

$$\mathbb{P}(\zeta_i \mid \theta) = \frac{1}{Z(\theta)} \exp(\theta^T \phi(\zeta_i)) = \frac{1}{Z(\theta)} \exp \left[\sum_{t=0}^{|\zeta_i|-1} \theta^T \phi(S_t) \right] \tag{2.8}$$

where $Z(\theta)$ is the partition function. MaxEnt-IRL maximizes the likelihood of the observed data under the maximum entropy (exponential family) distribution.

Adversarial Inverse Reinforcement Learning (AIRL) Based on MaxEnt-IRL [27], combine sample-based MaxEnt-IRL with forward reinforcement learning to estimate the partition function Z, where:

$$L(\theta) = -\mathbb{E}_{\zeta_i \sim p}\, r_\theta(\zeta_i) + \log \left(\mathbb{E}_{\zeta_j \sim q} \left[\frac{\exp(r_\theta(\zeta_j))}{q(\zeta_j)} \right] \right). \tag{2.9}$$

Here, $r_\theta(\zeta_i)$ is the reward of trajectory ζ_i, p represents the distribution of demonstrated samples, while q is the background distribution for estimating the partition function $\int \exp(r_\theta(\zeta)) d\zeta$. Due to high variance from operating over entire trajectories, Fu et al. [32] extend the algorithm to single state-action pairs and the proposed method, AIRL, which is a practical and scalable IRL algorithm based on an adversarial reward learning formulation. We use AIRL to recover the reward function for complex interactive systems since AIRL can estimate non-linear reward functions.

Distance Minimization Inverse Reinforcement Learning (DM-IRL) For completeness, we also employ DM-IRL [12, 25], which deals with scored trajectories, to have a case of perfectly recovered reward weight θ for comparison. DM-IRL directly attempts to regress the user's actual reward function that explains the given score. DM-IRL uses discounted accrued features to represent the trajectory:

$$\psi(\zeta_i) = \sum_{t=0}^{|\zeta_i|-1} \gamma^t \phi(S_t), \tag{2.10}$$

[4] IRL methods are described in greater detail in Section 2.2.2.

where γ is the discount factor. The score of a trajectory ζ_i is

$$\text{score}_{\zeta_i} = \theta^T \psi(\zeta_i). \tag{2.11}$$

Since the exact score for each trajectory is supplied, the recovered rewards with DM-IRL are exactly the ground truth of reward functions, which can be regarded as oracle rewards.

This process is depicted in Figure 2.1 by the arrow marked *recover*. Once we have recovered the reward function $r(s)$, we can proceed to the optimization objectives presented in Eq. 2.6.

2.6 Optimizing Interactive Systems

We start by explaining how to maximize the quality of an interactive system for a user behaving according to a fixed stationary policy π:

$$T^*_\pi = \arg\max_{T \in T} V^\pi_T. \tag{2.12}$$

To solve this problem, we first build an MDP as proposed above, where the user is the agent and the system is the environment. Following Assumption 2, the system can be optimized to improve the user experience, which we characterized by the quality of the interactive system. This problem is *equivalent* to finding the optimal policy in a reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$, where the agent is an interactive system and the stochastic environment is a user. It should be noted that the roles of agent and environment in the reformulated MDP$^+$ are exactly the opposite of the roles in the original MDP. We also convert the state space and action space correspondingly. We rely on the first MDP for inferring the user reward functions, while we rely on the second one, MDP$^+$, for updating interactive systems. In MDP$^+$, the state S^+_t is represented by a concatenation of the state S_t the user is in and the action A_t the user takes at time step t from the original MDP; the action A^+_t is the original state S_{t+1}. The interactive system observes the current state S^+_t and picks an action A^+_t under the interactive system policy $\pi^+(A^+_t|S^+_t)$. Then the user returns the next state S^+_{t+1} according to the transition distribution $T^+(S^+_{t+1}|S^+_t, A^+_t)$ which is inferred from the policy model $\pi(A_{t+1}|S_{t+1})$. Therefore, finding the optimal transition T^*_π from Eq. 2.12 is equivalent to finding the optimal policy π^+_* in the reformulated MDP$^+$ as follows:

$$\pi^+_* = \arg\max_{\pi^+ \in \Pi^+} V^{\pi^+}_{T^+}, \tag{2.13}$$

which can be done using an appropriate RL method such as Q-learning or Policy Gradient. D^+_0 is the initial distribution of states in MDP$^+$. After we have demonstrated how to optimize the interactive system for a given stationary policy, we return to the original problem of optimizing the interactive system for an optimal policy π_*.

To summarize, we propose a formal procedure for optimizing interactive systems, called ISO, presented in Algorithm 1, with the following steps:

Line 1 We assume that we have an estimate of the reward function $r(s)$ using one of the IRL methods described in Section 2.5.2. So we have as input: the original

Algorithm 1 Interactive System Optimizer (ISO)

1: **Input:** Original system (S, A, T), r, γ, D_0.
2: Construct original MDP(S, A, T, r, γ)
3: $\pi_*(a|s) = RL(S, A, T, r, \gamma)$ *// finding the current user policy*
4: Construct system MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$: *// reformulate the original MDP by switching the roles of agent and environment*
   - $S_t^+ = S_t \oplus A_t$ *// build the new state space by concatenation*
   - $A_t^+ = S_{t+1}$ *// build the new action space*
   - $T^+(S_{t+1}^+|S_t^+, A_t^+) = \pi_*(A_{t+1}|S_{t+1})$ *// build the transitions in* MDP^+
   - $r(S_t^+)^+ = r(S_t)$ *// convert the reward function*
   - $\gamma^+ = \gamma$ *// both MDPs share the same discount factor*
5: $D_0^+ \sim (S_0 \sim D_0, A_0 \sim \pi_*(a|S_0))$ *// sample initial states in* MDP^+
6: $\pi^+(A_t^+|S_t^+) = T(S_{t+1}|S_t, A_t)$ *// find the optimal transition distribution in the original MDP is formulated as fining the optimal policy in a reformulated* MDP^+
7: $\pi_*^+(a^+|s^+) = RL(S^+, A^+, T^+, r^+, \gamma^+)$ *// optimize the system policy in* MDP^+
8: $T^*(S_{t+1}|S_t, A_t) = \pi_*^+(A_t^+|S_t^+)$ *// replace the transition distribution in original MDP with the newly updated system policy*
9: **Output:** Optimized system (S, A, T^*)

system (S, A, T), the reward function r, the discount factor γ, and the initial distribution of states D_0.

Line 2 ISO formulates the original system as MDP(S, A, T, r, γ).

Line 3 ISO uses an appropriate RL algorithm to find the current user policy $\pi_*(a|s)$ given the reward function r.

Line 4 ISO transforms the original MDP(S, A, T, r, γ) into the new MDP$^+$ $(S^+, A^+, T^+, r^+, \gamma^+)$, where the roles of the agent and environment are switched. In our setting, S_t^+ has the same reward value as S_t. The discount factor γ^+ remains the same.

Line 5 ISO transforms D_0 to D_0^+ to match the distribution of first state-action pairs.

Line 6 The equivalence $\pi^+(A_t^+|S_t^+) = T(S_{t+1}|A_t, S_t)$ means that finding the optimal π_*^+ according to Eq. 2.13 is equivalent to finding the optimal T_π^* according to Eq. 2.12. Therefore, the transition distribution can be regarded as a policy network or a policy table from the MDP's perspective depending on the policy learning method.

Line 7 We can use an appropriate RL algorithm to find $\pi_*^+(A_t^+|S_t^+)$.

Line 8 ISO extracts $T^*(S_{t+1}|S_t, A_t)$ from the optimal system policy $\pi_*^+(A_t^+|S_t^+)$. The extraction process is trivial: $T^*(S_{t+1}|S_t, A_t) = \pi^+(A_t^+|S_t^+)$. Then, ISO terminates by returning the *optimized* interactive system.

Line 9 ISO outputs the optimized interactive system (S, A, T^*).

Once ISO has returned the optimized system (S, A, T^*), we expose it to users so they can interact with it as illustrated in Figure 2.1. We assume that users adjust their policy to T^*. After enough iterations, the user policy will converge to the optimal one. Iterations between optimizing the interactive system for the current policy and updating the user policy for the current interactive system continue until both converge.

In summary, we have presented the Interactive System Optimizer (ISO). It optimizes an interactive system using data-driven objectives. It works by transforming the original MDP, solving it, and using its solution to yield the optimal transition distribution in the original MDP.

2.7 Experiments and Results

In this section, we apply our proposed method, ISO, to two simulated interactive setups. In the first setup, the interactive system operates in a tabular-based world[5] with finite states and actions (Section 2.7.1). The second one has a more realistic setup, where the agent, the environment, and the reward function are all represented by separate neural networks (Section 2.7.2). Each proposed experimental setup is described using three components: (1) the design of the interactive system, (2) modeling user behavior, and (3) a suitable evaluation process. For both experimental setups, our results demonstrate that ISO can significantly improve the system performance in the designed setups. We conclude this section by discussing a list of limitations (Section 2.7.3).

2.7.1 Optimizing interactive systems in a tabular-based world

Experimental setup

Designing an interactive system We simulate an arbitrary interactive system where we need a finite set of states S, a finite set of actions A, and a transition distribution T. Features of a state $\phi(s)$ are fixed. For our experimental setup, we simulate an interactive system where $|S| = 64$ and $|A| = 4$. We work with a complex environment where a user can transition between any two states if these two states are connected. The connections between the two states are predefined and fixed, but the transition distribution is changeable. In words, for the same system in different runs, the connectivity graph of this system is fixed and will not be changed once it is sampled at the very beginning. This setup corresponds to the inherent constraints between state transitions in real interactive systems (Section 2.5.1). We use a hyper-parameter, the connection factor cf, to define the number of possible next states after the user has taken one specific action at the current state. For an initial interactive system, D_0 is randomly sampled as well as T. At each iteration, ISO delivers T^*, which substitutes the initial T obtained at the previous iteration. The *optimized* interactive system is used for the next iteration until the process converges.

[5] A tabular-based world is a two-dimensional, cell-based environment where the agent starts from one cell and moves toward the terminal cell while collecting as much reward as possible. The connections between different cells are predefined.

Modeling user behavior To model user behavior we require a true reward function $r_{real}(s)$, and an optimal user policy π^*_{user}. We utilize a linear reward function $r_{real}(s)$ by randomly assigning 25% of the states reward 1, while all others receive 0. As we use one-hot features for each state, $r_{real}(s)$ is guaranteed to be linear.

We use a soft value iteration method [163] to obtain the optimal user policy π^*_{user}.

The quality of the recovered reward functions is influenced by how trajectories are created, which in turn can affect the performance of ISO as it relies on $r_{real}(s)$ to optimize the transition distribution T behind the interactive system with reinforcement learning.

We experiment with the following types of user trajectories:

- *Optimal:* Users know how to behave optimally in an interactive system to satisfy their needs. To simulate the user interactions H, we use π^*_{user} trained with the real reward function $r_{real}(s)$.

- *SubOptimal:* Not all users know the system well, which means that the demonstrated behavior is a mixture of optimal and random. We propose two different methods to simulate suboptimal behavior. The degree of optimality of user behavior is controlled by either of two following factors: (1) the proportion of random behavior (this is called 'wandering' behavior in [146]); or (2) the user action noise, which is collectively called the noise factor (NF) $\in [0.0, 1.0]$.

  Mix of Behaviors (MB): The log of user interactions H is a mix of trajectories generated by the optimal policy and the adversarial policy.[6]

  Noise in Behavior (NB): In this case, the trajectories in H are generated from the optimal policy but we add noise to the user actions to get suboptimal behavior.[7]

The generated history of user interactions H represents the case of trajectories without a score which will be fed to MaxEnt-IRL. In terms of DM-IRL, interaction history should be given along with scores for each trajectory – $\hat{H}$. To generate the required dataset $\hat{H}$, we calculate the score using the true reward function $r_{real}(s)$. $\hat{H}$ is the input to DM-IRL.

At each iteration, we sample the following datasets reflecting different types of history of user interactions: $\hat{H}$, $H_{Optimal}$, $H_{SubOptimal-0.2-MB}$, $H_{SubOptimal-0.6-MB}$, $H_{SubOptimal-0.2-NB}$, $H_{SubOptimal-0.6-NB}$, each of size $15,000$ and $|\zeta_i| \in [30, 40]$.

Evaluation process To evaluate the performance of ISO, we report the *expected state value* under optimal policy Eq. 2.5 for an *initial* interactive system (S, A, T_{init}) and an *optimized* one (S, A, T_{opt}), which we derive after around 100 *iterations* (one *iteration* means we sequentially recover the reward function and run Algorithm 1 once). A higher

[6] To model suboptimal user behavior we use two user policies: (1) an optimal user policy π^*_{user}; and (2) an adversarial policy $(1 - \pi^*_{user})$, which means we choose the action that has the lowest likelihood according to π^*_{user}. We include an adversarial policy instead of a random one because it is the hardest case as users behave opposite of what we expect. E.g., $NF = 0.2$ means that 20% of the trajectories are generated with the adversarial policy.

[7] E.g., $NF = 0.2$ means the probability is 20% that the user will not choose the action with the highest probability in the optimal policy.

expected state value means users are more satisfied while interacting with the interactive system. We initialize 40 different initial interactive systems by randomly sampling reward functions and transition distribution, and report the overall performance over these 40 systems. The true reward functions and the connectivity graphs of these sampled systems are fixed in the whole optimization process. We use the recovered reward function with DM-IRL as the oracle reward for in this setup.

Results and discussion

Improving interactive systems with ISO Figure 2.2, 2.3, 2.4 show how the quality of the interactive system increases with each iteration of ISO in terms of different connection factors. The final relative improvements after optimization can be found in Table 2.1. We use IRL-labeled to represent the system optimized with the recovered reward function by method DM-IRL. As expected, when the user gives feedback about the quality of the trajectories (IRL-labeled), the task is simpler and ISO manages to get high improvements with the oracle rewards. However, the picture changes when we hide the scores from the trajectories. Without scores, ISO relies on the optimality of user behavior to recover the reward function. As the optimality decreases, so does the behavior of ISO, and the performance decays. With oracle rewards from DM-IRL, ISO converges quite fast – as we can see in Figure 2.3 and Figure 2.4, after 20 iterations the expected state value begins to plateau. Most improvements happen in the first several iterations. Thus, ISO works with accurately labeled trajectories, but usually obtaining high-quality scores is intractable and expensive in a real interactive system because the real rewards are invisible. We report it as the oracle performance in our experiment.

With respect to trajectories without scores, ISO is able to improve the initial expected state value. In Figure 2.2, the influence of the noise factor and types of trajectories (MB or NB) is clear. However, in Figure 2.4, where there are fewer connections between two states, only the convergence speeds of different curves are different but they all converge to the same state value eventually. ISO manages to optimize the interactive system even though the user trajectories are quite noisy.

More interestingly, the convergence speed and final converged values are different depending on the connection factors. As we can see, it is more difficult to get high performance when there are more connections between different states in the predefined systems. More connections mean that more possible trajectories could be taken and it is intractable for MaxEnt-IRL to learn a reward function from this kind of situation. In contrast, in Figure 2.4, each state-action pair can only have two possible next states and the final average state value is much higher than the system in Figure 2.2.

Impact of ISO components The performance of ISO depends on its two components: (1) RL methods used to optimize the user policy π_{user} for the original MDP and system policy π^{+}_{sys} for the reformulated MDP^{+}; and (2) IRL methods – to estimate the true reward function $r_{real}(s)$. The dependence on RL methods is obvious – the result will only be as good as the quality of the final optimization, so an appropriate method should be used. The performance of ISO can be influenced by the quality of the recovered reward functions, $r(s)$. For the case of labeled trajectories, the values of $r(s)$ recovered by DM-IRL are identical to the ground truth $r_{real}(s)$ since a regression model is used

Table 2.1: The performance of ISO, measured as a relative improvement (Impr.) in expected state value over the Initial interactive system of the Optimized version (after 120 and 90 iterations) for different types of user behaviors: (a) IRL-labelled, (b) Optimal, (c) SubOptimal-0.2-MB, (d) SubOptimal-0.6-MB, (e) SubOptimal-0.2-NB, (f) SubOptimal-0.6-NB. Only IRL-labelled has access to trajectory labels. * indicates statistically significant changes ($p < 0.01$) using a paired t-test over the initial expected state value and the optimized expected state value.

CF \ BT	(a) IRL-labelled			(b) Optimal (NF=0.0)		
	Initial	Optimized	Impr.	Initial	Optimized	Impr.
32	1.50	4.21	281%*	1.50	2.45	164%*
8	1.98	4.21	213%*	1.98	4.04	205%*
2	2.92	4.08	140%*	2.92	3.86	132%*

CF \ BT	(c) SubOptimal-0.2-MB (NF=0.2)			(d) SubOptimal-0.6-MB (NF=0.6)		
	Initial	Optimized	Impr.	Initial	Optimized	Impr.
32	1.50	2.52	169%*	1.50	1.92	128%*
8	1.98	3.94	200%*	1.98	3.41	173%*
2	2.92	3.83	131%*	2.92	3.71	127%*

CF \ BT	(e) SubOptimal-0.2-NB (NF=0.2)			(f) SubOptimal-0.6-NB (NF=0.6)		
	Initial	Optimized	Impr.	Initial	Optimized	Impr.
32	1.50	2.40	160%*	1.50	1.96	131%*
8	1.98	4.06	206%*	1.98	3.83	194%*
2	2.92	3.90	134%*	2.92	3.72	128%*

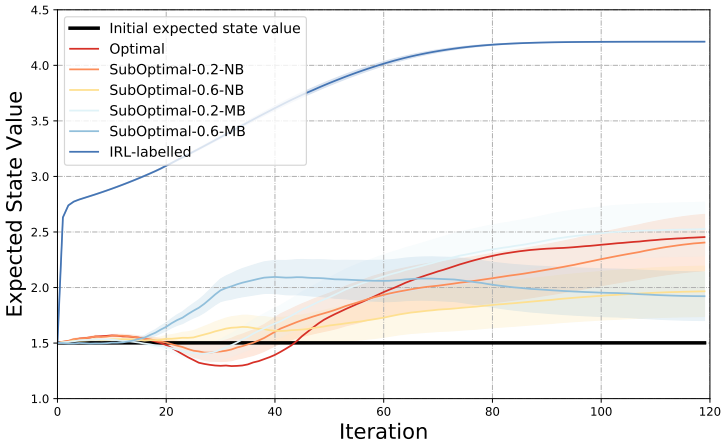


Figure 2.2: Performance of ISO over 40 randomly sampled systems when connection_factor=32. The error bounds denote the standard error of the mean (±SEM). The x-axis is the number of iterations of ISO and the y-axis is the expected state value.

and we have the exact score for each user trajectory. For the case of trajectories without scores, the quality of the recovered reward function is worse than DM-IRL. MaxEnt-IRL can only give a general overview of $r_{real}(s)$ if the user trajectories are optimal. If there are not enough constraints on the connections between states, with each iteration of running ISO, the shape of the sampled trajectories becomes more similar, which means that most trajectories pass by the same states and the diversity of trajectories decreases. We found that this makes it even more difficult to recover $r_{real}(s)$ and the MaxEnt-IRL quality deteriorates with the number of iterations, which results in lower performance in Figure 2.2. Hence, improving the performance of IRL methods is likely to significantly boost the performance of ISO and more advanced IRL methods could be adopted according to the real task.

2.7.2 Optimizing interactive systems in a network-based world

Experimental setup

Designing an interactive system with neural networks In this setup, we first present a simulated framework used for optimizing the interactive system (S, A, T) with ISO. Based on the two-step optimization setup in Section 2.6 we designed two separate optimizing modules respectively. Figure 2.5 shows the architecture of the optimizing module for the original MDP(S, A, T, r, γ), while Figure 2.6 describes the optimizing module for the reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$ respectively. As described in Section 2.6, we use MDP(S, A, T, r, γ) for reward learning and the reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$ for system optimization.

In the proposed setup, we have a continuous state space S and discrete action space A for the original MDP(S, A, T, r, γ), where the dimension of S is $S_{dim} = 50$ and action number is $|A| = 10$. The user policy π_{user}, the system policy π_{sys} and the

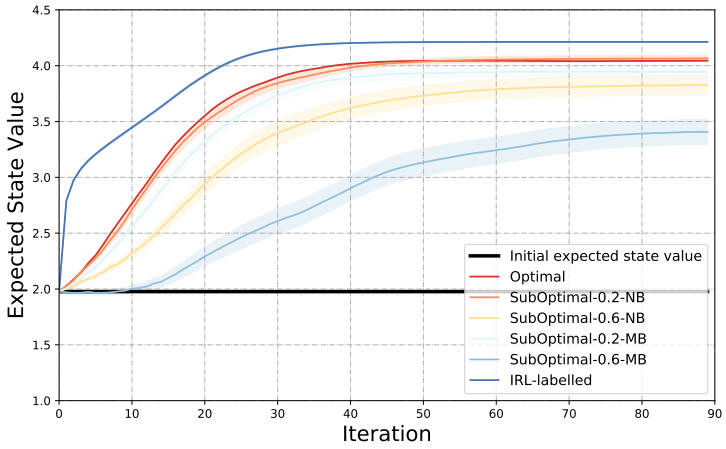


Figure 2.3: Performance of ISO over 40 randomly sampled systems when connection_factor=8. The error bounds denote the standard error of the mean (±SEM). The x-axis is the number of iterations of ISO and the y-axis is the expected state value.

reward function $r(s)$ are represented with multi-layer perceptrons separately. Following Algorithm 1, the transition distribution $T(S_{t+1}|S_t, A_t)$ is exactly the system policy π_{sys} which is fixed in this step. We assume the state distribution follows a multivariate Gaussian distribution with a diagonal covariance matrix and the system policy π^+_{sys} will produce the corresponding mean and variance. Since the state space S is continuous, the output of π_{sys} will be a sampled continuous state s_{t+1} at next step $t+1$ given s_t and a_t. Here we use Proximal Policy Optimization (PPO) [118], a policy gradient based method, to optimize the user policy π_{user}. With respect to the reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$, the state s_t and action a_t from the original MDP will be concatenated to form the new state s^+_t following Algorithm 1. The action a^+ is from continuous action space and the transition distribution $T^+(S^+_{t+1}|S^+_t, A^+_t)$ is exactly the user policy π_{user} in the original MDP. Different from π_{sys} in the original MDP, π^+_{sys} will be updated with PPO and it will be used to replace π_{sys} in the original MDP after optimization finished. $r^+(s)$ is the learned reward function in the first optimizing step.

With respect to the optimizing module for MDP(S, A, T, r, γ) shown in Figure 2.5, the user policy is wrapped up with a PPO agent and the reward function is loaded to the reward agent. To estimate the user reward function, we utilize Adversarial Inverse Reinforcement Learning (AIRL) in the reward learning step. The user policy agent and the reward agent make up the AIRL agent. As an adversarial learning method, the AIRL agent needs user traces generated by real users to update the reward function. In this setup, we use the user agent $\pi^*_{user\_real}$ trained with the true reward function $r_{real}(s)$ to produce necessary user-system interaction traces, which will be stored in the *Expert Behavior* area. The environment *Environment-1* for AIRL training and behavior generation mainly consists of the system policy π_{sys} to deliver the next state s_{t+1} given state s_t and action a_t according to $T(S_{t+1}|S_t, A_t)$. The system policy π_{sys} will keep fixed in MDP(S, A, T, r, γ). It should be noted that there are two different user reward functions in Figure 2.5. The reward function $r_{airl}(s)$ in AIRL agent is updated during

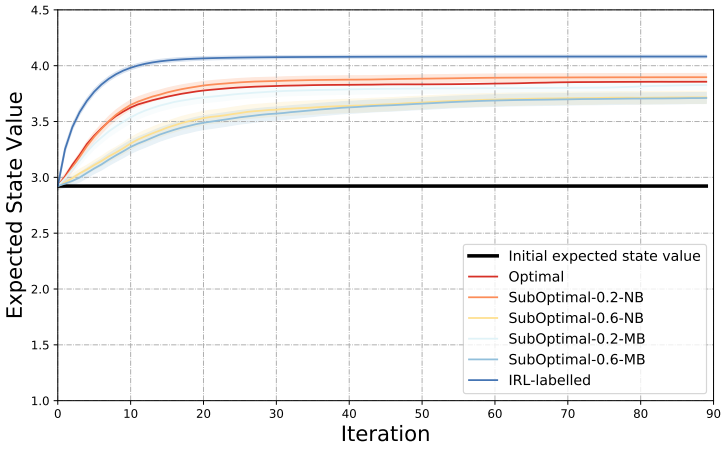


Figure 2.4: Performance of ISO over 40 randomly sampled systems when connection_factor=2. The error bounds denote the standard error of the mean (±SEM). The x-axis is the number of iterations of ISO and the y-axis is the expected state value.

AIRL training while the reward function $r_{real}(s)$ in the expert agent is the true reward function. The AIRL agent and system policy has no access to the true reward function $r_{real}(s)$ and we use $r_{airl}(s)$ to approximate $r_{real}(s)$, which is also the motivation of AIRL.

The optimizing module for the reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$ shown in Figure 2.6 is responsible of updating the system policy $\pi_{sys}$with the recovered reward function $r_{airl}(s)$. Just like other reinforcement learning setups, it has three main components: an environment, a PPO agent, and the reward function. The system policy π_{sys} is wrapped up with a PPO agent and the reward agent is the function $r_{airl}(s)$ learned in MDP(S, A, T, r, γ). Given state s_t^+ and action a_t^+, the step function of the environment *Environment-2* will return the next state s_{t+1}^+ according to $T^+(S_{t+1}^+|S_t^+, A_t^+)$ in Line 4 of Algorithm 1, where the user policy π_{user} is involved.

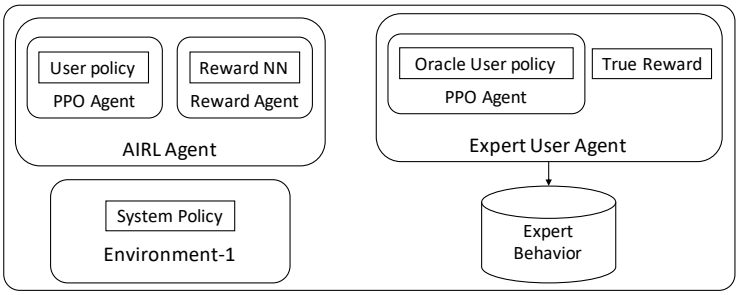


Figure 2.5: The architecture of the optimizing module in the original MDP(S, A, T, r, γ), which is responsible for generating user behavior and recovering user reward functions.

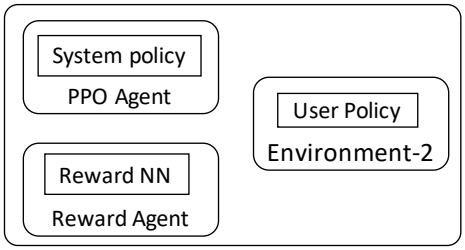


Figure 2.6: The architecture of the optimizing module in the reformulated MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$, responsible for optimizing the system agent.

Modeling user behavior Given the current system policy π_{sys} and the real user reward function $r_{real}(s)$, we optimize the user policy π_{user} by running PPO method. The optimized user policy will be saved as the oracle user policy $\pi^*_{user\_real}$. Then by making the user policy $\pi^*_{user\_real}$ interact with the system policy π_{sys}, we can collect a bunch of interaction trajectories ($20K$ in our experiments) and all these behavior data will be loaded to the expert behavior bucket. The maximum length of the collected trajectories is 40. The stored user interaction traces will be used to learn the user reward function $r_{airl}(s)$ (we use AIRL method in this setup).

Evaluation process To evaluate the performance of ISO in the proposed framework, we report the *Average Return* of m sampled trajectories ($m = 1000$ in our setup) under optimal policy $\pi^*_{user\_real}$ under the real reward $r_{real}(s)$ for an *initial* interactive system and an *optimized* one, which we derive after 3 *iterations* (one *iteration* means we sequentially recover the reward function and run Algorithm 1 once). A higher average return means users are more satisfied while interacting with the interactive system. We initialize 5 different initial interactive systems by randomly sampling the system policy π_{sys}, and report the overall performance over these 5 systems. Besides, we want to avoid the situations that the optimized system has totally different behaviors compared to the initial system because the dramatic change may hurt users' experience. To make sure the systems before and after optimized follow similar behaviors, we introduce a regularization term, the KL-Divergence $\lambda * D_{KL}(T_{opt} \mid T_{init})$, to control the distance between these two system policies. This term can also be regarded as the inherent constraints between state transitions, just like the "connection factor" in Section 2.7.1. The hyperparameter λ is applied to control the effect of the term. Due to the training complexity of network-based simulations, we run 5 times for each parameter setup rather than 40 times in the tabular world.

The ground truth of user reward functions With respect to the true reward function r_{real}, we have two different setups: *a handcrafted reward function* and *a randomly initialized reward function*. For the handcrafted one, we use $r_{real}(s) = \frac{1}{s_{dim}}\phi(s) * \phi(s)$ as the reward for the given state s. In terms of the sampled reward function, we initialize the parameters of the reward network with uniform distributions, and this makes recovering the reward function more difficult because there are no patterns in the sampled reward function. In the real world, users always have their preferences and the

reward function in users' minds is not likely to be random. The true reward function $r_{real}(s)$ is fixed in the whole optimizing process.

Results and discussion

In this section, we first discuss the results of the experiments with a manually designed real reward function. Then, we move to the discussion of the experimental results with randomly initialized reward function.

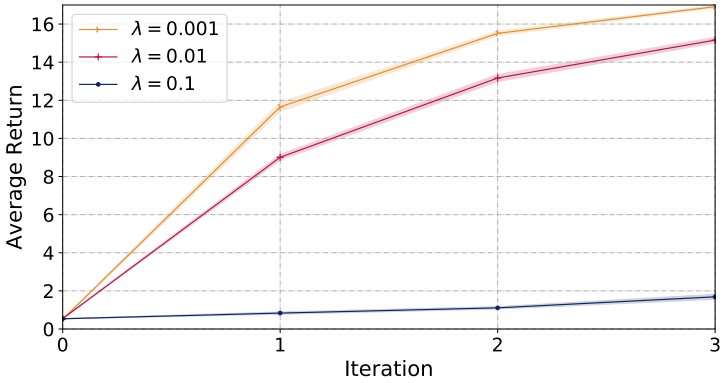


Figure 2.7: The state performance during optimization with oracle reward function and oracle user policy. The real reward function is manually designed. The error bounds denote the standard error of the mean (±SEM).

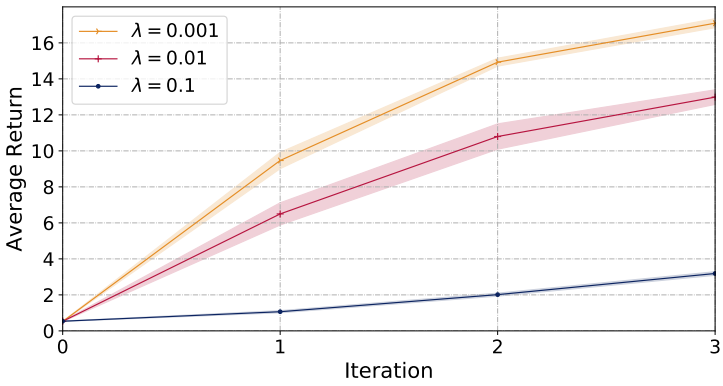


Figure 2.8: The state performance during optimization with recovered reward and oracle user policy. The real reward function is manually designed. The error bounds denote the standard error of the mean (±SEM).

Manually designed real reward function To verify if the proposed two-MDP framework works or not, we first skip the reward learning step and use the oracle reward

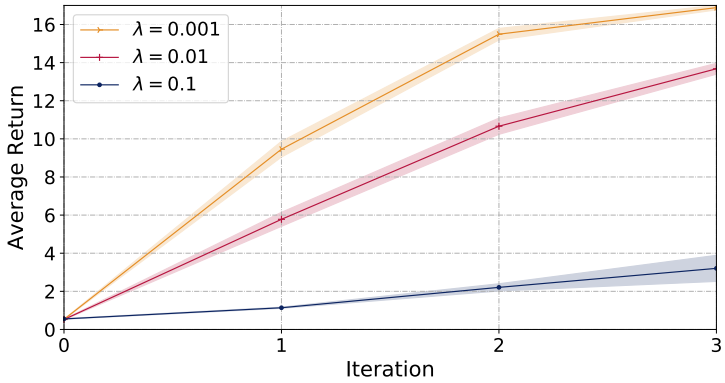


Figure 2.9: The state performance during optimization with recovered reward function and recovered user policy. The real reward function is manually designed. The error bounds denote the standard error of the mean (±SEM).

function $r_{real}(s)$ as the "learned" reward function with collected user behaviors. With respect to the user policy π_{user} used to interact with the system agent in the second optimizing module, we use the oracle user policy $\pi^*_{user\_real}$ trained with true reward function $r_{real}(s)$. Other modules stay the same and we obtain the performance in Figure 2.7. An interactive system at iteration 0 is the initial system and not optimized yet. As we can see, with a looser restriction (i.e., a smaller λ value) on the distance between the optimized system and the initial system, we can achieve higher performance with respect to the average trajectory returns. After we bring back the reward learning step and use the learned reward function $r_{airl}(s)$ to optimize the system policy, we have the results shown in Figure 2.8. The system can still achieve higher performance by running Algorithm 1. If we compare the results between systems $\lambda = 0.001$ in Figure 2.7 and Figure 2.8, we can find that the system trained with oracle reward $r_{real}(s)$ can hit higher returns after two iterations. The finding still holds with respect to the systems $\lambda = 0.01$ in both setups. However, this is not the case when we set $\lambda = 0.1$. We suspect this is because the large regularization term $D_{KL}(T_{opt} \mid T_{init})$ has brought too many uncontrollable factors into the optimization step, which may disturb the training.

As mentioned in Section 2.7.2, the user policy π_{user} is essential while optimizing the system π_{sys}. In Algorithm 1, the user policy π_{user} plays the role of the transition distribution T^+ in the environment *Environment-2*. In addition to the two reward function setups above, we need to conduct an experiment with the user policy π_{user} trained with recovered reward function $r_{airl}(s)$ for system optimization in the reformulated MDP$^+$. Since we use AIRL to learn the reward function in this framework, we have the estimated user policy $\pi^*_{user\_airl}$ which is rebuilt during the adversarial training process. We replace $\pi^*_{user\_real}$ in the environment *Environment-2* with this rebuilt policy $\pi^*_{user\_airl}$. In terms of the reward function r^+ in MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$, we use the reward function $r_{airl}(s)$. By running Algorithm 1, we have the results in Figure 2.9. It is clear that the rebuilt policy $\pi^*_{user\_airl}$ can still help with improving the system performance. This is meaningful because by using adversarial training we can rebuild

the user policy and user reward function simultaneously. The accuracy of the estimated user policy will definitely benefit from a high-quality estimation of the user reward function. The only moment that real users are involved happens when we are collecting user-system interaction trajectories. This perfectly matches the scenarios in real life, where we first collect interaction histories from users and then infer the user preferences (r_{airl}) and user behavior patterns (π_{user_airl}) according to the collected data. In the next step, the system policy π_{sys} will be optimized based on user preferences and user behavior patterns. In the end, the newly updated system (S, A, T^*) will be presented to users to improve their user experience. If necessary, new interaction trajectories will be collected and another optimization turn can start again.

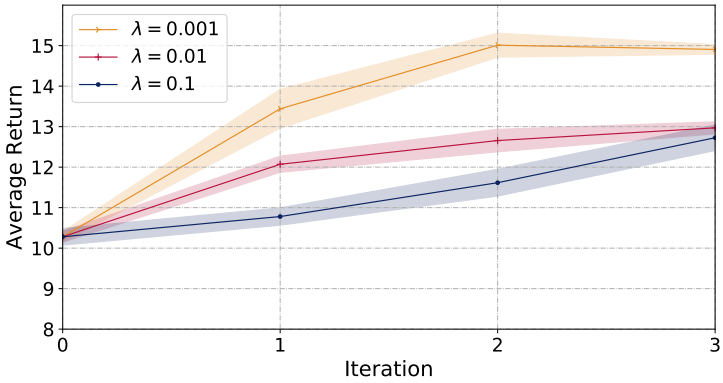


Figure 2.10: The state performance during optimization with oracle reward function and oracle user policy. The real reward function is randomly initialized. The error bounds denote the standard error of the mean (±SEM).

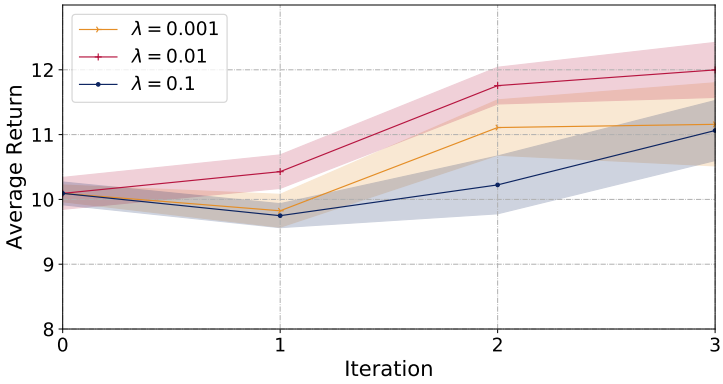


Figure 2.11: The state performance during optimization with recovered reward and oracle user policy. The real reward function is randomly initialized. The error bounds denote the standard error of the mean (±SEM).

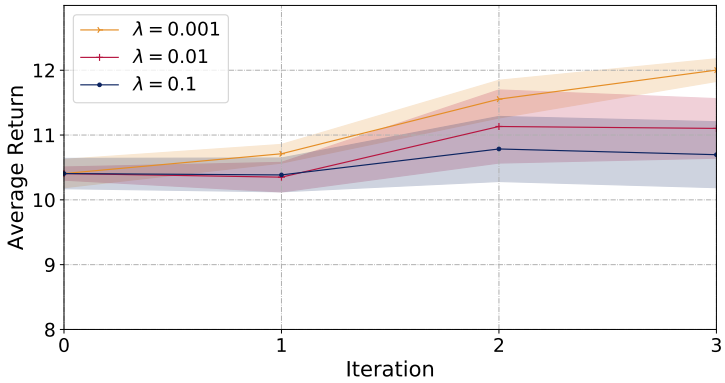


Figure 2.12: The state performance during optimization with recovered reward function and recovered user policy. The real reward function is randomly initialized. The error bounds denote the standard error of the mean (±SEM).

Randomly initialized reward function In this section, we show how the interactive optimizer performs when the reward function $r_{real}(s)$ is randomly initialized. In Figure 2.10, with the real reward function $r_{real}(s)$, the system can still achieve relatively large improvements in terms of average return. The fact that all curves have higher starting points is because the randomly initialized system policy has the advantage to hit higher reward for a random reward function and this will not hold when the reward function has a special pattern as in Section 2.7.2. We also find that looser restrictions on the distance between the optimized system and the initial system can bring larger performance improvements, as we observed in Section 2.7.2.

With respect to Figure 2.11 and Figure 2.12, the improvements still exist but are not so significant compared to those with the handcrafted reward function in Figure 2.8 and Figure 2.9. A potential reason is that it is hard to recover a high-quality reward function given user behaviors generated by a random reward function. Especially before the first iteration, the system still performs randomly (the initial system is randomly initialized) and it is difficult to collect useful interaction traces for reward learning, and this is also the reason why, in Figure 2.11 and Figure 2.12, the average returns of some curves even drop after the first iteration. However, in the real world, users always have their preferences and the reward function in users' minds is unlikely to be random. Besides, the initial system will not behave randomly because in most cases a real interactive system (e.g., search engine, digital assistant) will be tested offline first and will not be deployed before it can achieve reasonable performance. This will alleviate the reward learning stress to some degree. We have this random reward function here simply to validate how well the method could perform with most uncontrollable behaviors.

2.7.3 Limitations

First, to recover a reliable reward function, a large number of high-quality user interaction traces are essential, which can come with a great cost in real life (but is not impossible). Furthermore, an interactive system usually serves different users,

which can lead to a violation of Assumption 5 about the homogeneity of user behavior. Therefore, we would need to work on personalizing the recovered reward functions. A possible way to address this limitation in the future is to incorporate the user features into the state space, but this still needs to be explored.

Second, as shown in Section 2.7.1, the final performance of the optimized system highly relies on the quality of the recovered reward function. With respect to the more advanced extension of Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL), which is Adversarial Inverse Reinforcement Learning (AIRL), the adversarial training process is intractable for complex real behavior. The two limitations above boil down to the quality of recovered reward functions, given limited user traces in real life. Third, after we have inferred the reward function, we will update the system in a reformulated MDP setup, where we switch the roles between the agent and environment. The potential problem that can arise is that the action space for the new MDP could be extremely large and this may present challenges for the scalability of the Reinforcement Learning (RL) process.

Finally, we validate our method in two simulated experimental setups. Although we try to design our setups as close as possible to the real-word scenarios, there is a potential gap between the designed systems and real-world applications. But the positive verification of our method in a simulated setup helps us to better understand the pros and cons of the proposed approach and help us with planning the experiments with the real-world scenarios in the near future.

To summarize, we have proposed two experimental setups to test the proposed framework: tabular and neural. In both cases, the results demonstrate significant improvements in interactive systems when applying ISO. We conclude this section acknowledging a number of possible limitations, some of which can be considered as future directions.

2.8 Conclusions and Future Work

We have recognized that previous work on interactive systems has relied on the assumption that handcrafted objective functions can accurately reflect users' preferences and intentions while interacting with interaction systems. As a result, interactive systems have been optimized for manually designed objectives that do not always align with the true user preferences and cannot be generalized across different domains. To overcome this discrepancy, we have proposed a novel two-step framework to optimize interactive systems, which first infers the user reward model given collected user interaction traces and then updates the system with the inferred reward functions via a novel algorithm: the Interactive System Optimizer (ISO).

Firstly, we modeled user-system interactions using MDP, where the agent is the user, and the stochastic environment is the interactive system. User satisfaction is modeled via rewards received from interactions, and the user interaction history is represented by a set of trajectories. We followed the previously justified assumption that user incentive to interact with the system if they are rewarded. Treating an interactive system as a changeable and programmable environment is novel and reasonable because we have complete control of the interactive systems since we are the system designers.

Secondly, we formalized an optimization problem to infer user needs from observed

user-system interactions, in the form of a data-driven objective. Importantly, our method works without any domain knowledge and is thus even applicable when prior knowledge is absent.

Thirdly, we proposed a novel, Interactive System Optimizer (ISO), that iterates between optimizing the interactive system for the current inferred objective; and letting the user adapt to the new system behavior. This process repeats until both the user and system policies converge. Our experimental results show that ISO robustly improves user satisfaction.

Given the solutions above and the experiment results, it is obvious that the answer to research question RQ1 is *"Yes"*. The newly proposed solution to optimize an interactive system based on data-driven objectives is novel, many promising directions for future work are possible. For instance, while ISO performs well for users with a single goal, this approach could be extended to settings with multiple goals. Similarly, extensions considering more personalized goals could benefit the overall user experience. Finally, investigating the scalability and real-world applicability of ISO could open many research possibilities.

In the following chapter, we will start the experiments with more realistic applications, dialogue systems. The interactive system optimizer proposed in this chapter is more like a theoretical guideline. We will verify the effectiveness of data-driven objectives in optimizing the dialogue generation model, which is a typical interactive system.

3 Dialogue Generation: From Imitation Learning to Inverse Reinforcement Learning

In this chapter, we investigate two adversarial training methods for open-domain dialogue systems. They can be regarded as practical applications of data-driven objectives. We conduct different experiments to answer the research question:

> *Can data-driven reward functions be used to successfully improve open-domain dialogue systems?*

3.1 Introduction

The task of an open-domain dialogue system is to generate sensible dialogue responses given a dialogue context [68, 69, 108, 123, 154]. There are two broad directions for training a dialogue generation system: the first employs defined rules or templates to construct possible responses and the second builds a chatbot to learn the response generation model with a machine translation framework from social dialogue collections [120, 121, 123, 128]. Sequence-to-sequence (Seq2Seq) models enjoy the advantages of scalability and language independence and the maximum likelihood estimation objectives make it simple to train them. However, in dialogue generation, the trained model suffers from generating dull and generic responses such as "I don't know" [66, 69, 120, 128], which are meaningless. Li et al. [66] suggest that "by optimizing for the likelihood of outputs given inputs, neural models assign a high probability to 'safe' responses". To alleviate this problem, Li et al. [68] introduce a neural Reinforcement Learning (RL) generation method to generate coherent and interesting dialogues by optimizing the manually defined reward function covering ideal dialogue properties. However, a handcrafted reward function is expensive to maintain and does not generalize over different domains [27, 32]. Especially for open-domain dialogue systems, it is hard to decide what knowledge is essential to design a proper reward function [69]. Additionally, the accuracy of defined reward functions can degrade when the dialogue context becomes more complex. Li et al. [69] use adversarial training for

This chapter was published as [75].

dialogue generation, where they jointly train two systems, a generative model to produce response sequences and a discriminator to distinguish between the human-generated dialogues and the machine-generated ones. Feedback from the discriminator is used as a reward to push the generator to produce more realistic replies. The discriminator takes a dialogue consisting of a context-reply pair as input and outputs the probability that this dialogue is from real human dialogues.

In Li et al. [69], during generator training, the reward of each generated word during decoding should be supplied and the Monte Carlo search is applied to estimate the reward for each word position. A potential problem is that the returned reward from the discriminator could be very sparse and unstable, which may lead the generator to produce unintended and nonsense replies. Moreover, Li et al. [69] put no constraints on the generator policy, which can result in two problems. First, the learned policy may prefer to generate general responses. Second, the training step can easily get stuck in a local optimum, which leads the generator to produce identical responses regardless of the input context or even worse – the outputs from the generator are always the same ungrammatical sentence.

In this chapter, we first extend the adversarial dialogue generation method introduced by Li et al. [69] to a new model, DG-AIL, which incorporates an entropy regularization term to the generation objective function. This addition can alleviate the problem of mode collapse. Then we adopt adversarial inverse reinforcement learning to train a dialogue generation model, DG-AIRL. This method enables us to both make use of an efficient adversarial formulation and recover a more precise reward function for open-domain dialogue training. Unlike Shi et al. [125], we design a specific reward function structure to measure the reward of each word in generated sentences while taking account of the dialogue context. We also consider two human evaluation settings to assess the overall performance of our model.

To summarize, we make the following contributions:

- A novel reward model architecture to evaluate the reward of each word in a dialog, which enables us to have a more accurate signal for adversarial dialogue training;
- A novel Seq2Seq model, DG-AIRL, for addressing the task of dialogue generation built on adversarial inverse reinforcement learning;
- An improvement of the training stability of adversarial training by employing causal entropy regularization;

3.2 Background

Preliminaries

As in Chapter 2, we build our dialogue system as a Markov Decision Process (MDP), which is defined by a tuple (S, A, τ, r, γ), where S and A are the state space and action space, respectively, τ is the transition probability, and $\tau(s, a, s')$ is the probability of transitioning from state s to state s' under action a at time t:

$$\tau(s' \mid s, a) = P(s_{t+1} = s' \mid s_t = s, a_t = a). \tag{3.1}$$

Here, $r(s, a)$ is the immediate reward after taking action a in state s; $\gamma \in [0, 1]$ is a discount factor.

The *dialogue response strategy* is represented by a policy, which is a mapping $\pi \in \Pi$ from states $s \in S$ and actions $a \in A$ to $\pi(a|s)$, which is the probability of performing action $a_t = a$ by the user when in state $s_t = s$:

$$\pi(a|s) = P(a_t = a \mid s_t = s). \tag{3.2}$$

Maximum causal entropy

Motivated by the task of decision prediction in sequential interactions, Ziebart et al. [165] propose to use maximum causal entropy to model the availability and influence of sequentially revealed side information. The causal entropy of policy π is defined as:

$$H(\pi) \triangleq E_{\pi}[-\log \pi(a|s)]), \tag{3.3}$$

which measures the uncertainty presented in policy π [165].

Maximum entropy inverse reinforcement learning (MaxEnt-IRL)

Given a set of demonstrated (expert) behavior, which can be seen as the trajectories resulting from executing expert policy π_E, Inverse Reinforcement Learning (IRL) aims to find a reward function that can rationalize the given behavior. In Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) [164], the demonstrated behavior $D_{demo} = \{\zeta_1, \ldots, \zeta_N\}$ is assumed to be the result of an expert acting stochastically and near-optimally with respect to an unknown reward function. Trajectories with equivalent rewards have equal probability to be selected and trajectories are sampled from the distribution:

$$p(\zeta_i \mid \theta) = \frac{1}{Z(\theta)} \exp^{r_\theta(\zeta_i)} = \frac{1}{Z(\theta)} \exp^{\sum_{t=0}^{|\zeta_i|-1} r_\theta(s_t, a_t)}, \tag{3.4}$$

where $Z(\theta) = \int \exp(r_\theta(\zeta))d\zeta$ is the partition function and r_θ is the reward function, which takes a state-action pair as input. MaxEnt-IRL maximizes the likelihood of the demonstrated data D_{demo} under the maximum entropy (exponential family) distribution and the objective is given as:

$$L(\theta) = -\,\mathbb{E}_{\zeta \sim D_{demo}}\, r_\theta(\zeta) + \log Z. \tag{3.5}$$

This task can be seen as a classification problem where each trajectory represents one class. However, it is difficult to apply vanilla MaxEnt-IRL to complex and high-dimensional settings since computing the partition function $Z(\theta)$ is intractable in the original method. To overcome this drawback, Finn et al. [27] combine sample-based maximum entropy IRL with forward reinforcement learning to estimate the partition function Z, where:

$$L(\theta) = -\,\mathbb{E}_{\zeta_i \sim p}\, r_\theta(\zeta_i) + \log \left(\mathbb{E}_{\zeta_j \sim q} \left[\frac{\exp(r_\theta(\zeta_j))}{q(\zeta_j)} \right] \right). \tag{3.6}$$

Here, p represents the distribution of demonstrated samples, while q is the background distribution for estimating the partition function $\int \exp(r_\theta(\zeta))d\zeta$. This work alternates between updating the reward function r_θ to maximize the likelihood of the demonstrated data and optimizing the background distribution q to minimize the variance of the importance sampling estimation.

Generative adversarial imitation learning

Recovering the true reward function is intractable in real scenarios [32, 41, 164]. In previous research, if only the optimal policy is pursued, imitation learning is used to rebuild the policy network directly by skipping recovering reward functions. Ho and Ermon [41] cast the problem of IRL as an optimization problem in the paradigm of Generative Adversarial Networks (GANs), where the discriminator corresponds to the reward function and the generator corresponds to the policy used to sample trajectories. The optimization problem is given as:

$$\max_{r \in R} \left(\min_{\pi \in \Pi} -\lambda H(\pi) - \mathbb{E}_{\pi}[r(s,a)] \right) + \mathbb{E}_{\pi_E}[r(s,a)]. \tag{3.7}$$

The optimization of Eq. 3.7 is converted to an imitation learning algorithm:

$$\min_{\pi \in \Pi} -\lambda H(\pi) + D_{JS}(\rho_{\pi}, \rho_{\pi_E}), \tag{3.8}$$

which finds a policy π whose occupancy measure ρ_{π} minimizes the Jensen-Shannon divergence to the expert's policy π_E (the policy of demonstrated data). The occupancy measure ρ_{π} can be interpreted as the unnormalized distribution of state-action pairs that an agent encounters when navigating the environment with policy π. Eq. 3.8 can be solved by finding a saddle point (π, D) of the expression

$$\mathbb{E}_{\pi}[-\log(D(s,a))] + \mathbb{E}_{\pi_E}[-\log(1 - D(s,a))] - \lambda H(\pi), \tag{3.9}$$

where D is a binary classifier to distinguish state-action pairs of π and π_E.

3.3 Method

In this section, we will first extend the work by Li et al. [69] to the framework of adversarial imitation learning, and then introduce our main model, which applies adversarial inverse reinforcement learning to train a dialogue system.

3.3.1 Problem setting

In a dialogue setting, the word sequence $\langle w_1, w_2, \ldots, w_t \rangle$ in an utterance can be regarded as corresponding actions $\langle a_1, a_2, \ldots, a_n \rangle$ taken by the policy network at different time steps. We use a state function f to compress the dialogue context and the words already generated in the current utterance to an intermediate representation, which will be regarded as the current state. For example, $s_0 = f(p)$ represents the state at time step 0 and it takes the dialogue context p as input. State s_t is given as $s_t = f(p, a_1, a_2, \ldots, a_{t-1})$. In this work, we limit the range of the dialogue context to the utterances in the last two conversation turns.

Given an initial state s_0 representing the history of previous dialogues, a well-trained dialogue system should reply with a reasonable sentence $\langle w_0, w_1, \ldots, w_t \rangle$ generated by selecting a specific word at different time steps. The length t is automatically decided by the policy network. We aim to find the optimal policy $\pi(a_t|s_t)$ that selects the most appropriate word at each time step.

3.3.2 Dialogue generation with adversarial imitation learning (DG-AIL)

In the framework of adversarial imitation learning, we aim to train a dialogue system to imitate the way humans talk by observing real human dialogues. This model DG-AIL can be regarded as an extension of the work of Li et al. [69].

Unlike previous work, we do not only consider the difference between the distributions of real dialogues and generated dialogues but also take into account how the previous state-action pairs affect future words under a specific policy network π, which can be measured by the causal entropy $H(\pi)$.

In adversarial learning, the task of the discriminator D is to distinguish dialogues from the true data distribution and dialogues from the generator. As shown in Figure 3.1, we adopt a hierarchical structure to represent the discriminator model. The first layer is an input encoder that compresses the utterances from each speaker in the conversation. Then, a context encoder sequentially takes as input the utterance representations and generates a final state to represent the whole dialogue. In the end, the final state is fed to a binary classifier that predicts whether the dialogue is real or fake with a confidence value.

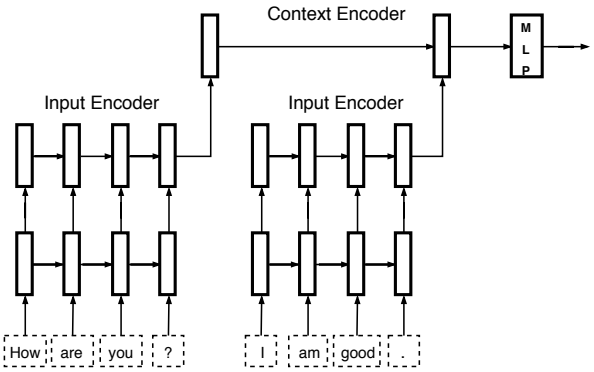


Figure 3.1: Discriminator architecture in DG-AIL.

According to Eq. 3.9, the gradient of the discriminator parameters is given as:

$$\nabla D_\theta = \mathbb{E}_{\zeta\sim\pi}[\nabla_\theta \log(D_\theta(s,a))] + \mathbb{E}_{\zeta\sim\pi_E}[\nabla_\theta \log(1 - D_\theta(s,a))]. \tag{3.10}$$

The generative model G attempts to generate high-quality human-like responses to confuse the discriminative classifier D while maintaining a high policy entropy. The gradient to update the generator parameters can be inferred from Eq. 3.9 as follows:

$$\begin{aligned} \nabla G_\phi &= \nabla_\phi[-\lambda H(\pi_\phi) - \mathbb{E}_{\zeta\sim\pi_\phi}[D_\theta(s,a)]] \\ &= -\mathbb{E}_{\zeta\sim\pi_\phi}\nabla_\phi[\log(\pi_\phi(a|s))](Q_\theta(s,a) - \lambda \log \pi_\phi(a|s)) \end{aligned} \tag{3.11}$$

where $Q_\theta(s,a) = \mathbb{E}_\zeta[\log(D_\theta(s,a))|s_0 = \bar{s}, a_0 = \bar{a}]$ is estimated with Monte Carlo search.

3.3.3 Dialogue reward learning with adversarial inverse reinforcement learning (DG-AIRL)

Our main model DG-AIRL adopts inverse reinforcement learning techniques to train a dialogue generation model. We assume that human participants in a dialogue are using a true reward function that guides them to formulate a policy to react with different replies to different dialogue contexts. Unlike the use of a classifier to supply a reward signal in the model DG-AIL, the reward model in DG-AIRL has a more specific architecture to evaluate the reward for each state-action pair, which can provide more accurate and precise reward signal to update the generator.

Dialogue response policy

In MaxEnt-IRL, the reward model (discriminator) attempts to assign high rewards to demonstrated trajectories (from the expert policy) and low rewards to sampled trajectories from other policies. In this way, when the reward function is fixed, the expert policy can be found by solving a common reinforcement learning problem:

$$G_\phi(r_\theta) = \arg\min_{\pi \in \Pi} -\lambda H(\pi) - \mathbb{E}_{\zeta \sim \pi}[r_\theta(\zeta)], \tag{3.12}$$

where ζ represents the sampled dialogues and $H(\pi)$ is the causal entropy regularization term; $r_\theta(\zeta)$ is the reward of dialogue ζ that can be accessed from the reward model. The goal of the generator is to generate dialogues that can achieve higher rewards from the reward model. The found policy maximizes the expected cumulative reward while maintaining high-entropy.

The derivative can be inferred as follows:

$$\begin{aligned} \nabla_\phi G(r)_\phi &= \nabla_\phi[-\lambda H(\pi_\phi) - \mathbb{E}_{\zeta \sim \pi_\phi}[r_\theta(\zeta)]] \\ &= -\mathbb{E}_{\zeta \sim \pi_\phi} \nabla_\phi[\log(\pi_\phi(\zeta))](r_\theta(\zeta) - \lambda \log \pi_\phi(\zeta)). \end{aligned} \tag{3.13}$$

If we decompose dialogue ζ into different time steps, the gradient is given as:

$$\begin{aligned} \nabla_\phi G(r)_\phi &= -\mathbb{E}_{\zeta \sim \pi_\phi} \nabla_\phi[\log(\pi_\phi(\zeta))](r_\theta(\zeta) - \lambda \log \pi_\phi(\zeta)) \\ &= -\textstyle\sum_t \mathbb{E}_{\pi_\phi(a_t|s_t)} \nabla_\phi[\log(\pi_\phi(a_t \mid s_t))](r_\theta(\zeta_{t:T}) - \lambda \log \pi_\phi(a_t \mid s_t)). \end{aligned} \tag{3.14}$$

The reward $r_\theta(\zeta_{t:T})$ of a partial dialogue from time t to T is estimated with Monte Carlo search.

Reward learning

Following prior work on sample-based maximum entropy IRL (Eq. 3.6), the objective (loss function) of our reward model is given as:

$$L(\theta) = -\mathbb{E}_{\zeta_i \sim \pi_E} r_\theta(\zeta_i) + \log \mathbb{E}_{\zeta_j \sim \pi} \left(\frac{\exp(r_\theta(\zeta_j))}{q(\zeta_j)} \right), \tag{3.15}$$

where π_E denotes the policy of demonstrated trajectories and π the policy of background samples. The term q denotes the background distribution from which dialogues ζ_j were

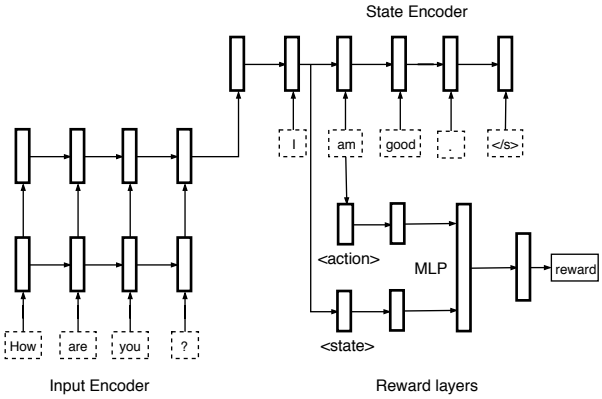


Figure 3.2: Reward model architecture in DG-AIRL.

sampled. In our setting, q is the distribution of dialogues generated with the current dialogue policy π. We use D_{demo} and D_{samp} to represent the set of dialogues generated with policy π_E and π, respectively.

The gradient of the reward function is given by:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{\zeta_i \in D_{demo}} \nabla_\theta r_\theta(\zeta_i) + \frac{1}{Z} \sum_{\zeta_j \in D_{samp}} w_j \nabla_\theta r_\theta(\zeta_j), \tag{3.16}$$

where $w_j = \frac{\exp(r_\theta(\zeta_j))}{q(\zeta_j)}$ and $Z = \sum_j w_j$.

As shown in Figure 3.2, our reward model in DG-AIRL consists of two RNN encoders and one MLP network. The input encoder compresses the utterances from the context into a context representation which becomes the initial state in the next step. The state encoder takes as input the dialogue context and generated words before time t and outputs the new state representation s_t for time step t. Then the state and action representations are fed to two separate MLP layers respectively. The outputs of these two models are concatenated and form the input to the third MLP layer to get the final reward value of current state-action pair $\langle s_t, a_t \rangle$.

3.4 Experimental Setup

3.4.1 Dataset

The MovieTriples dataset [120] has been developed by expanding and preprocessing the Movie-Dic corpus [6] of film transcripts and each dialogue consists of 3 turns between two interlocutors. The dialogues are collected from the scripts of more than 600 movies, which span a wide range of topics. We limit the length of the utterances from one speaker in each dialogue turn between 4 and 80. In the final dataset, there are around 157,000 dialogues in the training set, 19,000 in the validation set, and 19,000 in the test set. The average length of each dialogue is about 54.

3.4.2 Experimental settings

We limit the vocabulary table size to the top 20k most frequent words for the MovieTriples dataset. All words that are not in the vocabulary tables are replaced with the token "$\langle unk \rangle$". Following the preprocessing method from [120], all names and numbers are replaced with the "$\langle person \rangle$" and "$\langle number \rangle$" tokens, respectively [108]. Since the context input in each dialogue is made up of several utterances from two different speakers, to capture the interactive structure, we insert a special token "$\langle /s \rangle$" between the first turn and the second. The word embedding size is 200.

Next, we list the models we consider. We implement all models based on Tensorflow[1] except VHRED.

DG-AIRL This is our main model that adopts adversarial inverse reinforcement learning techniques to train a dialogue system. The encoder and decoder in the generator (policy network) are built from a 2-layer GRU with 1024 hidden units and an attention mechanism is incorporated into the decoding step. With respect to the reward function structure, we choose a 2-layer GRU with 1024 hidden units as the context encoding layer to compress the input to an intermediate representation. Then a 1-layer GRU with 1024 hidden units is used to take state-action pair as input and output the next state as shown in Fig.3.2.

Seq2Seq The encoder and decoder in this baseline are copied from the generator in the DG-AIRL model and built from a 2-layer GRU with 1024 hidden units; an attention mechanism is incorporated into the decoding step.

SeqGan This is the model from [69]. In terms of the generator, SeqGan shares the same architecture as the DG-AIRL model. With respect to the discriminator in this model, both the first encoder layer and the second context layer are built from a 2-layer GRU with 1024 units separately.

VHRED For the VHRED model, we reuse the original implementation from the authors, including their tuning techniques.[2]

DG-AIL This is the model with the adversarial imitation learning method, which is also an extension of SeqGan. The DG-AIL model shares the same structure as the SeqGan model, including generator and discriminator. The only difference is the loss function, as we discussed in Section 3.3.

We optimize the models using Adam [52] and the learning rate is initialized as 0.001 except for VHRED. Dropout with probability 0.3 was applied to the GRUs and we apply gradient clipping for both policy models and reward models. We set the beam size to 8 for Monte Carlo search during training and beam search during testing. During the training of SeqGan, DG-AIL, and DG-AIRL, we employ the teacher-forcing technique from Li et al. [69] to increase training efficiency.[3]

[1] `https://www.tensorflow.org/`

[2] For more details, see `https://github.com/julianser/hed-dlg-truncated`.

[3] The source code of this work is available at `https://bitbucket.org/ZimingLi/dg-irl-aaai2019`

3.4.3 Evaluation metrics

To evaluate the response quality in dialogue generation, recent work has adopted word-overlap metrics from machine translation to compare a machine-generated response to a single target response [33, 67, 98, 109]. Since the response to the input context in dialogue could be very diverse and open, a single target response is not able to cover all reasonable answers. Liu et al. [84] show that word-overlap metrics such as BLEU [98] correlate very weakly with reply quality judgments from human annotators. To assess the performance of our proposed algorithm, we use two evaluation methods, one is to use word embedding based metrics and the other is to employ human annotators to judge the response quality. We also tried to evaluate the response diversity with the metric *Distinct* but we found that the result is not aligned with the results based on human evaluations; we do not include results based on *Distinct* in this chapter.

Embedding metrics

With respect to word embedding based methods, we use three metrics that are also used in [121]:

- *Average embedding*: This method applies cosine-similarity to measure the similarity between the mean word embeddings of the target utterance and the predicted utterance.
- *Greedy embedding*: This metric relies on cosine-similarity but adopts greedy matching to find the closest word in the target response for each word in the generated response [111].
- *Extrema embedding*: This method computes the word embedding extrema scores [30] that embed the responses by taking the extrema (maximum of the absolute value) of each dimension, and afterward computes the cosine similarity between them.

A higher score indicates that the generated reply shares similar semantic content with the target response. For all three metrics, we use pre-trained Word2Vec word embeddings trained on the Google News Corpus, which is public access[4].

Human evaluation

Proper quality evaluation of dialogue responses should cover not only topic-similarity but also lexical aspects, informativeness, interestingness, and so on. There is currently no reliable metric to assess the overall quality of dialogue responses. For this reason, we create human annotations to evaluate the quality of responses given dialogue context with a crowdsourcing platform[5]. Previous work involving human evaluation usually has two experimental settings: pairwise comparison and pointwise scoring. In pairwise comparisons, annotators are asked to choose the better response from replies generated by two models while in the pointwise method, annotators are asked to rate the overall quality of each response, typically on a scale from 0 (low quality) to 4 (high quality). We employ both pairwise and pointwise assessments. We use a pairwise setting to directly contrast the overall performance of our model against others. Pointwise scoring may be noisier than pairwise judgments since human annotators need to give an exact score.

[4] `https://code.google.com/archive/p/word2vec/`
[5] FigureEight: `https://www.figure-eight.com/`

However, pointwise judgments give us a chance to analyze the differences between replies at a fine-grained level of detail.

We randomly sample 1,000 dialogue contexts from the test set of the MovieTriple dataset. Each context has five replies from five generation models and we have 5,000 context-reply pairs in total; 2,500 are used for pointwise scoring while the remaining 2,500 are grouped into 2,000 comparison pairs for the pairwise setting. Each comparison pair has one dialogue context and two replies, where one is from our DG-AIRL model and the other is from a baseline model.

For the pairwise setting, we ask annotators to judge which of the two responses is more appropriate given a dialogue context. We instruct annotators which aspects they should take into account when making a decision. The top priority is that an appropriate response must be relevant; besides, they should consider:

- whether the response is natural;
- whether the response is interesting;
- whether the response can make the conversation continue, that is, whether the response is proactive; and
- whether the response is the only possible reply to the given context.

If the annotators think neither of the responses is more appropriate or it is impossible to infer the conversation from the given context, they are asked to choose the third choice – "Neither is more appropriate". We insert test questions to exclude annotators who lack the capacity to finish the tasks, such as limited English skills. We only accept annotators considered "highly trusted" by the crowdsourcing platform and require 90% accuracy on designed "test questions". Each comparison pair is assessed by three annotators.

For pointwise judgments, annotators were asked to judge the overall quality from 0 to 2:

+2: The response is not only relevant and natural but also informative and interesting; the response need not be so interesting, but it is natural and can make the conversation continue (more proactive); the response is the only possible reply to the dialogue.

+1: The response can be used as a reply to the context, but it is too generic like "I don't know". These replies are usually more reactive.

0: The response cannot be used as a reply to the context. It is either semantically irrelevant or disfluent.

At the start of the annotation effort, we instruct the annotators and show them several examples of how to assign grades to a given dialogue. We use the same quality checks and annotator selection criteria as in the pairwise setting. Each context-reply is assigned to three human annotators.

3.5 Results and Analysis

3.5.1 Results using embedding metrics

In Table 3.1, we report the scores obtained using the embedding metrics (Section 3.4.3). All response generation models are fine-tuned to obtain the highest score on the validation dataset. We found 0.01 and 0.1 to be the optimal values of λ for the DG-AIL

Table 3.1: Performance in terms of embedding metrics of response generation models, with 95% confidence intervals. * indicates the result is statistically significant ($p < 0.005$) with a paired t-test over DG-AIRL and other baseline models.

Model	Average	Greedy	Extrema	Length
Seq2Seq	0.563 ± 0.003	0.167 ± 0.001	0.352 ± 0.002	8.8
SeqGan	0.564 ± 0.003	0.165 ± 0.001	0.354 ± 0.002	9.7
VHRED	0.507 ± 0.003	0.145 ± 0.001	0.309 ± 0.002	**12.0**
DG-AIL	0.553 ± 0.003	$\mathbf{0.171^* \pm 0.001}$	0.356 ± 0.002	7.7
DG-AIRL	$\mathbf{0.589^* \pm 0.003}$	0.169 ± 0.001	$\mathbf{0.368^* \pm 0.002}$	10

model and the DG-AIRL model.

The DG-AIRL and DG-AIL models achieve the highest scores using the embedding metrics, which means that they can better capture the topic of the target response than other models.

The performance of the VHRED model is unexpected since this method achieves the lowest value while it is one of the state-of-the-art methods in dialogue response generation. A possible reason is that the other four models adopt an attention mechanism to directly capture the relation between generated words and input words from context. Serban et al. [121] state that VHRED produces longer responses and its responses are on average more diverse based on unigram entropy. Besides, DG-AIL and DG-AIRL are also supposed to generate more diverse responses. However, a more diverse response does not mean it is an appropriate response. If the response deviates too much from the target topic, the response content will not be relevant to the dialogue context and it deserves a lower quality score. On the other hand, if a diverse response is appropriate to the dialogue context, it is unfair to use embedding-based metrics to assess these kinds of generative models.

Given these considerations, we believe that embedding-based metrics may not be powerful enough to reflect the overall response quality and that it is essential to carry out human evaluations.

3.5.2 Results using human annotations

Pairwise evaluation

As shown in Table 3.2, our model DG-AIRL outperforms other response generation models based on pairwise comparisons. Among the first three models, the DG-AIRL model outperforms them all at the probability 0.46.

The difference is that the **Win** rate of VHRED (the lose rate of DG-AIRL) is lower than Seq2Seq and SeqGan. In other words, although VHRED has a higher probability to be tied with DG-AIRL, it loses more compared to Seq2Seq and SeqGan.

As we said in the last section, a possible explanation for these results is that VHRED does produce longer responses (Table 3.1) but the contents of these responses deviate too much from the target topic, which could result in lower performance. In our human evaluation setting, fluency is not the only aspect annotators need to consider while determining their preference for a response. By taking into account different factors,

Table 3.2: Performance in terms of pairwise human annotations of response generation models.

Model pair	Win	Tie	Loss
DG-AIRL-Seq2Seq	**0.44**	0.29	0.27
DG-AIRL-VHRED	**0.46**	0.32	0.22
DG-AIRL-SeqGan	**0.47**	0.25	0.28
DG-AIRL-DG-AIL	0.36	**0.37**	0.27

Table 3.3: Performance in terms of pointwise human evaluations of response generation models. "Freq of N" is the relative frequency of a model's responses with a score of N.

Model	Freq of +2	Freq of +1	Freq of 0	Avg Score
Seq2Seq	0.09	0.22	0.69	0.40
SeqGan	0.09	0.21	0.70	0.39
VHRED	0.12	0.25	0.63	0.49
DG-AIL	0.12	0.29	0.59	0.53
DG-AIRL	0.13	0.28	0.59	**0.54**

such as relevance, fluency, informativeness, we think the final judgments from human annotators are trustworthy. The Fleiss' kappa score, which indicates the agreement among labelers [29], is around 0.23. This value is not high and a possible reason is that judging the response quality is challenging for human annotators when only 1 or 2 utterances are provided as context, especially on the MovieTriples dataset.

Compared to the first three models, the DG-AIL model achieves a better performance. According to the performance of DG-AIL and SeqGan, we can claim that causal entropy regularization improves the performance of dialogue models that employ adversarial training because they share the same structure and the only difference is the entropy regularization term. This also echoes the use of entropy regularization in adversarial models [42]. Although the DG-AIRL model still beats DG-AIL, the difference between the **Win** and the **Loss** rates is much smaller compared to other models. DG-AIL model has a higher chance to draw with the DG-AIRL model. These two models both adopt entropy regularization and they have the same generator structure. The performance difference comes from the reward model: DG-AIRL is able to improve the response quality because it has a specific reward model for each state-action pair and adopts importance sampling. The reward signal in DG-AIRL is more concrete and reliable compared to DG-AIL.

Pointwise evaluation

The evaluation results based on pointwise judgments are shown in Table 3.3. According to the average score, the DG-AIRL and DG-AIL models outperform other models, and the performance of DG-AIRL and DG-AIL is quite close. This does not mean that the result of the pointwise evaluation conflicts with the conclusion we made in the last section, viz. that the DG-AIRL model beats the DG-AIL model based on the pairwise

Table 3.4: Example output. People names are replaced with the token "$\langle person \rangle$". All words are converted to lowercase.

Context	Next Reply from Speaker A
Speaker A: we did a story on this guy . he owns half of arizona . **Speaker B:** is he a fraud ?	**Reference:** how would you ever know ? **Seq2Seq:** yeah . no . i do not know who he is . **SeqGan:** sure , i guess not . **VHRED:** he doesn't want to know . **DG-AIL:** no . no , he is not . **DG-AIRL:** no . maybe he is a good man .
Speaker A: and i am honored that you want to be with me because you would never be with anyone who was not truly worthy . **Speaker B:** i feel exactly the same way about you .	**Reference:** oh, $\langle person \rangle$, do not say that , please , that just makes it worse . **Seq2Seq:** do you love your $\langle unk \rangle$. **SeqGan:** can we i . you . you . you . you . you . you . . you . **VHRED:** $\langle person \rangle$? **DG-AIL:** what is it . **DG-AIRL:** is that what you think .
Speaker A: oh , that was not my father . **Speaker B:** what ?	**Reference:** that was not him . i heard him tell the cops . **Seq2Seq:** i am sorry . i needed to know how i felt about it . **SeqGan:** you said he was a singer . **VHRED:** i can't . i'm sorry . **DG-AIL:** i do not know . i guess . **DG-AIRL:** it is not him . i do not know what happened .

evaluation. Compared to the pairwise evaluation, the pointwise evaluation needs to assign an exact score to each context-reply pair and this score is independent of the other replies to the same dialogue context. In contrast, in the pairwise evaluation, we consider a pair of replies to the same context at the same time and it is more natural and reliable if we want a ranked list based on performance. The advantage of the pointwise setting is that it can provide quality distributions of different models and help us find out what makes a model performance different.

As shown in Table 3.3, we find that VHRED, DG-AIL, and DG-AIRL generate almost the same number of high-quality responses (responses that received a score "+2"). The VHRED model loses the competition with DG-AIRL and DG-AIL models because it generates more low-quality replies (responses that get score "0"). In our experimental setup, we ask annotators to grade the reply quality as "+1" (fine quality) if the response can be used as a reply to the message, but is too generic. In Section 3.1, we expect to generate more diverse responses and avoid producing too generic responses,

such as “I don’t know”. However, in some dialogue contexts, “I don’t know” is still an appropriate and reasonable response. As shown in Table 3.4, DG-AIRL, and DG-AIL improve the proportion of high-quality responses without losing the capacity to generate fine quality replies.

3.6 Related Work

Based on developments in sequential neural networks, Shang et al. [123] and Sordoni et al. [128] propose to generate high-quality replies in a dialogue system with a recurrent neural network. To formulate the complex dependencies between different utterances in multi-turn dialogs, Serban et al. [120] propose to adopt a hierarchical recurrent encoder-decoder neural network (HRED) to the dialogue domain, where word-level and utterance-level Recurrent Neural Networks (RNN) are used.

Built on HRED, the same group of authors create a more powerful generative architecture [121] with latent stochastic variables that span a variable number of time steps (VHRED). To train these RNN models, supervised training is commonly used, which minimizes the cross-entropy between the generated reply and an oracle reply. However, in terms of open-domain dialogue systems, there could be multiple reasonable replies for the same input context. In other words, the entropy of the target replies is high.

Li et al. [69] cast the task of open-domain dialogue generation as an RL problem and train a generator based on the signal from a discriminator to generate response sequences indistinguishable from human-generated dialogs.

3.7 Conclusion

In this chapter, we have investigated two adversarial training methods for open-domain dialogue systems. We have first adopted adversarial imitation learning to force our model to generate human-like dialogue responses. Besides that, we have incorporated an entropy regularization term to the generator objective function, which can alleviate the problem of mode collapse. Our second and main method, DG-AIRL, relies on techniques of adversarial inverse reinforcement learning. We design a specific reward architecture to supply a more accurate and precise reward signal for the generator training.

To assess the overall performance of our models, we propose two human-evaluation settings. We adopt the results from a pairwise evaluation setting to show that our model can outperform state-of-the-art methods in open-domain dialogue generation. To analyze the differences in replies from different models, we explore the results from a pointwise evaluation setting, which can provide a general quality distribution for different models.

In terms of the answer to the research question RQ2, we can confirm that data-driven reward functions can help with building high-quality dialogue generation models. With respect to the future work, it is promising to extend the idea of reward learning to multi-turn dialogue generation, which can propagate the reward signal between conversation turns. Another promising research direction is to explore the usefulness of recovered

reward models, for instance, to evaluate the quality of generated responses from other models.

In this chapter, we apply the policy gradient-based algorithm to update the dialogue policy. This is the only choice to utilize data-driven reward functions in the schema of adversarial training. In the next chapter, we will explore if it is possible to make off-policy based algorithms (e.g., DQN[91]) also benefit from data-driven reward functions in the area of task-oriented dialogue systems.

4

Guided Dialogue Policy Learning without Adversarial Learning in the Loop

This chapter is aimed at answering the following research question:

> *Can off-policy Reinforcement Learning (RL) methods benefit from data-driven objectives in dialogue policy learning for Task-oriented dialogue systems (TDSs)?*

We decompose the vanilla adversarial training used for dialogue policy learning into two sequential steps and apply it to task-oriented dialogue systems with off-policy reinforcement learning methods.

4.1 Introduction

Task-oriented dialogue systems (TDSs), such as Siri, Google Assistant, and Amazon Alexa, aim to offer users assistance with completing tasks. TDSs need dialogue policies to select appropriate actions at each dialogue step according to the current context of the conversation [13]. The development of Reinforcement Learning (RL) in robotics and other domains has brought a new view on learning dialogue policies [34, 130, 147]: it allows us to train with far more data than can be feasibly collected from actual users. The aim of Task-oriented dialogue system (TDS) is to maximize positive user feedback. TDS based on RL are amenable to training with user simulators instead of real humans [70, 115]. User simulators rely on a reward function that scores system actions given dialogue context [21, 101, 129, 149].

The most straightforward way to design a dialogue reward function is to score the agent based on the dialogue status in a rule-based fashion: if the dialogue ends successfully, a large positive reward will be returned; if the dialogue fails, the reward will be a large negative value; if the dialogue is still ongoing, a small negative value will be returned to encourage shorter sessions [101]. However, the rule-based solution is inflexible as it assigns the same negative reward to all the system actions before the dialogue ends. The sparse reward makes the qualities of different actions indistinguishable. Additionally, the rule-based approaches only return a meaningful reward when dialogue finishes, which can delay the penalty for low-quality actions and a high

This chapter was published as [78].

reward for high-quality ones during the conversation itself. Liu and Lane [83] address the difficulties listed above by employing adversarial training for policy learning by jointly training two systems: (1) a policy model that decides which action to take at each turn, and (2) a discriminator that marks if a dialogue was successful or not. Feedback from the discriminator is used as a reward to push the policy model to complete a task indistinguishably from humans. Improving upon this solution, Takanobu et al. [133] propose to replace the discriminator with a reward function that acts at the dialogue action level and returns the reward for the given action relying on the dialogue state, system action, and next dialogue state as its input. However, the described methods are limited to policy gradient-based algorithms, such as REINFORCE [152] and Proximal Policy Optimization (PPO) [118], to alternatively update the dialogue policy and the reward model on the fly, while off-policy methods are not able to benefit from self-learned reward functions. Furthermore, an alternating training schema for the dialogue agent and the reward model can easily get stuck in local optima or result in mode collapse.

To alleviate the problems mentioned above, in this chapter we propose a new approach for training dialogue policy by decomposing the adversarial learning method into two sequential steps. First, we learn the reward function using an auxiliary dialogue state generator where the loss from the discriminator can be backpropagated to the generator directly. Second, the trained discriminator as the dialogue reward model will be incorporated into the RL process to guide dialogue policy learning and will not be updated, while the state generator is discarded. Therefore, we can utilize any RL algorithm to update the dialogue policy, including both on-policy and off-policy methods. Additionally, since the reward function is pre-trained in an offline manner, we can first infer common information contained in high-quality human-generated dialogues by distinguishing human-generated dialogues from machine-generated ones, and then make full use of the learned information to guide the dialogue policy learning in a new domain in the style of transfer learning.

To summarize, our contributions are:

- A reward learning method that is applicable to off-policy RL methods in dialogue training.
- A reward learning method that can alleviate the problem of local optima for adversarial dialogue training.
- A reward function that can transfer knowledge learned in existing domains to a new dialogue domain.

4.2 Related Work

RL methods [21, 71, 75, 82, 99, 131, 149], have been widely utilized to train a dialogue agent by interacting with users. The reward used to update the dialogue policy is usually from a reward function predefined with domain knowledge and it could become very complex, e.g., in the case of multi-domain dialogue scenarios. To provide the dialogue policy with a high-quality reward signal, Peng et al. [100] propose to make use of the adversarial loss as an extra critic in addition to shape the main reward function. Inspired by the success of adversarial learning in other research fields, Liu and Lane [83] learn the reward function directly from dialogue samples by alternatively updating

the dialogue policy and the reward function. The reward function is a discriminator aiming to assign a high value to real human dialogues and a low value to dialogues generated by the current dialogue policy. In contrast, the dialogue policy attempts to achieve higher rewards from the discriminator given the generated dialogue. Following this solution, Takanobu et al. [133] replaces the discriminator with a reward function that acts at the dialogue action level, which takes as input the dialogue state, system action, and next dialogue state and returns the reward for the given dialogue action.

The key distinction of our work in this chapter compared to previous efforts is being able to train dialogue agents with both: (1) off-policy methods in adversarial learning settings; (2) the on-policy based approaches while avoiding potential training issues, such as mode collapse and local optimum. We propose to train (1) a reward model and (2) dialogue policy *consecutively*, rather than *alternatively* [83, 133]. To train the reward model, we introduce an auxiliary generator that is used to explore potential dialogue situations. The advantage of our setup is the transfer from a SeqGAN [156] to a vanilla GAN [37]. In a SeqGAN setup, the policy gradient method is essential to deliver the update signal from the discriminator to the dialogue agent. In contrast, in the vanilla GAN, the discriminator can directly backpropagate the update signal to the generator. Once we restore a high-quality reward model, we update the dialogue agent using common RL methods, including both on-policy and off-policy.

4.3 Learning Reward Functions

In this section, we introduce our method to learn reward functions with an auxiliary generator.

4.3.1 Dialogue state tracker

We reuse the rule-based ConvLab dialogue state tracker [61] to keep track of the information emerging in the interactions, including the informable slots that show the constraints given by users and requestable slots that indicates what users request. A belief vector is maintained and updated for each slot in every domain.

Dialogue state The collected information from the dialogue state tracker is used to form a structured state representation $state_t$ at every time step t. The final representation is formed by (1) the embedded results of returned entities for a query, (2) the availability of the booking option with respect to a given domain, (3) the state of informable slots, (4) the state of the requestable slot, (5) the last user action, and (6) the repeated times of the last user action. The final state representation S is a binary vector with 392 dimensions.

Dialogue action Each atomic action is a concatenation of domain name, action type and slot name, e.g., *Attraction_Inform_Address*, *Hotel_Request_Internet*. Since in the real scenarios, the response from a human or a dialogue agent can cover a combination of atomic actions, we extract the most frequently used dialogue actions from the human-human dialogue collections to form the final action space – A. For example,

[Attraction_Inform_Address, Hotel_Request_Internet] is regarded as a new action that the policy agent can execute. The final size of A is 300. We utilize one-hot embeddings to represent the actions.

4.3.2 Exploring dialogue scenarios with an auxiliary generator

We aim to train a reward function that can distinguish high-quality dialogues from unreasonable and inappropriate ones. To generate negative samples, we use an auxiliary generator *Gen* to explore the possible dialogue scenarios that could happen in real life. The dialogue scenario at time t is a pair of a dialogue state s_t and the corresponding system action a_t. The dialogue state-action pairs generated by *Gen* are fed to the reward model as negative samples. During reward training, the reward function can benefit from the rich and high-quality negative instances generated by the advanced generator Gen to improve the discriminability. Next, we will explain how states and actions are simulated, and our setup for adversarial learning.

Action simulation

To simulate the dialogue actions, we use a Multilayer Perceptron (MLP) as the action generator Gen_a followed by a Gumbel-Softmax function with 300 dimensions, where each dimension corresponds to a specific action from the defined A. The Gumbel-Max trick [38] is commonly used to draw a sample u from a categorical distribution with class probabilities p:

$$u = one\_hot(\arg\max_i[g_i + \log p_i]), \tag{4.1}$$

where g_i is independently sampled from Gumbel $(0, 1)$. Since the $\arg\max$ operation is not differentiable, no gradient can be backpropagated through u. Instead, we employ the soft-argmax approximation [46] as a continuous and differentiable approximation to $\arg\max$ and to generate the k-dimensional sample vector y following:

$$y_i = \frac{\exp((\log(p_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(p_j) + g_j)/\tau)}, \tag{4.2}$$

for $i = 1, \ldots, k$. When the temperature $\tau \to 0$, the $\arg\max$ operation is exactly recovered but the gradient will vanish. In contrast, when τ goes up, the Gumbel-Softmax samples are getting similar to samples from a uniform distribution over k categories. In practice, τ should be selected to balance the approximation bias and the magnitude of gradient variance. In this chapter, p corresponding to the output distribution of generator Gen_a and k equals to the action dimension 300.

State simulation

Compared to the GAN scenarios in computer vision, the output of the generator in our setting is a discrete vector which makes it challenging to backpropagate the loss from discriminator to the generator directly. To address this problem, we propose to project the discrete representation x in the expert demonstrations to a continuous space with

an encoder *Enc* from a pre-trained variational autoencoder [53]. We assume that the human-human dialogue state s is generated by a latent variable z_{vae} via the decoder *Dec* $p(s|z_{vae};\psi)$. Then we can regard the variable z_{vae} as the desired representation in a continuous space. Given a human-generated state s, the VAE utilizes a conditional probabilistic encoder Enc to infer z_{vae} as follows:

$$z_{vae} \sim \textit{Enc}(s) = q_\omega(z_{vae}|s), \tag{4.3}$$

where ω and ψ are the variational parameters encoder and decoder respectively. The optimization objective is given as:

$$L_{vae}(\omega,\psi) = \underbrace{\mathbb{E}_{z_{vae}\sim q_\omega(z_{vae}|s)}[\log p_\psi(s|z_{vae})]}_{(1)} + \underbrace{\textit{KL}(q_\omega(z_{vae}|s)||p(z_{vae}))}_{(2)}, \tag{4.4}$$

where (1) is responsible for encouraging the decoder parameterized with ψ to learn to reconstruct the input x; (2) is the KL-divergence between the encoder distribution $q_\omega(z_{vae}|s;\omega)$ and a standard Gaussian distribution $p(z_{vae}) = N(0, I)$.

The benefit of projecting the state representations to a new space is directly simulating the dialogue states in the continuous space $S_{\textit{embed}}$ similar to generating realistic images in computer vision. Besides, similar dialogue states are embedded into close latent representations in the continuous space to improve the generalizability. Figure 4.1 shows the overall process of learning the state projecting function $\textit{Enc}_\omega(s)$ given dialogue states from real human-human dialogues. We use $s_{\textit{real}}$ to denote the continuous representation of real state s while $s_{\textit{sim}}$ for the simulated one.

Adversarial training

We can approximate the real state-action distribution in a differentiable setup (1) by applying Gumbel-Softmax to simulate actions $a_{\textit{sim}}$; and (2) by directly generating simulated states $s_{\textit{sim}}$ in the continuous space $S_{\textit{embed}}$. The auxiliary generator $\textit{Gen}_\theta$ to simulate $s_{\textit{sim}}$ and $a_{\textit{sim}}$ has following components:

$$\begin{aligned} h &= \textit{MLP}_1(z_{sa}) \\ a_{\textit{sim}} &= f_{\textit{Gumbel}}(\textit{MLP}_2(h)) \\ s_{\textit{sim}} &= \textit{MLP}_3(h) \\ (s,a)_{\textit{sim}} &= s_{\textit{sim}} \oplus a_{\textit{sim}}, \end{aligned} \tag{4.5}$$

where θ denotes all the parameters in the generator and $\oplus$ is the concatenation operation. During the adversarial training process, the generator $\textit{Gen}_\theta$ takes noise z_{sa} as input and outputs a sample $(s,a)_{\textit{sim}}$ and it aims to get higher reward signal from the discriminator D_ϕ. The training loss for the generator $\textit{Gen}_\theta$ can be given as:

$$L_G(\theta) = -\mathbb{E}_{(s,a)_{\textit{sim}}\sim \textit{Gen}_\theta}(R_\phi((s,a)_{\textit{sim}}), \tag{4.6}$$

where $R_\phi((s,a)_{\textit{sim}}) = -\log(1 - D_\phi((s,a)_{\textit{sim}})$ and D_ϕ denotes the discriminator measuring the reality of generated state-action pairs $(s,a)_{\textit{sim}}$.

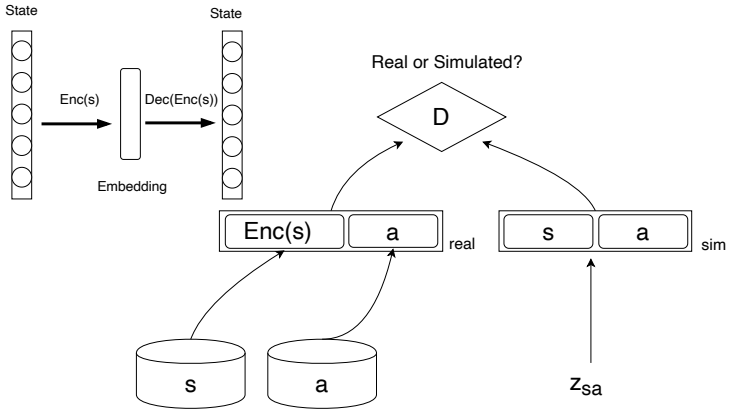


Figure 4.1: The architecture to simulate state-action representations with a variational autoencoder. z_{sa} is the sampled Gaussian noise.

The discriminator D_ϕ in this chapter is an MLP that takes as input the state-action pair (s, a) and outputs the probability $D(s, a)$ that the sample is from the real data distribution. Since the discriminator's goal is to assign a higher probability to the real data while lower scores to simulated data, the objective can be given as the average log probability it assigns to the correct classification. Given an equal mixture of real data samples and generated samples from the generator Gen_θ, the loss function for the discriminator D_ϕ is:

$$\begin{aligned} L_D(\phi) =& \mathbb{E}_{((s,a)_{sim})\sim Gen_\theta}(\log(1 - D_\phi((s, a)_{sim}))) \\ & - \mathbb{E}_{(s,a)\sim data}(D_\phi(Enc_\omega(s), a)_{real})). \end{aligned} \tag{4.7}$$

After the adversarial training is finished, we will keep the discriminator D_ϕ as the reward function for future dialogue agent training while the generator Gen_θ will be discarded.

Next, we discuss a suitable experimental environment for validating the presented method.

4.4 Experimental Setup

4.4.1 Dataset and training environment

MultiWOZ is a multi-domain dialogue dataset spanning 7 distinct domains[1], and $10,438$ dialogues [11]. The main scenario in this dataset is that a dialogue agent is trying to satisfy the demand from tourists such as booking a restaurant or recommending a hotel with specific requirements. The average number of turns is 8.93 and 15.39 for single and multi-domain dialogues, respectively.

[1] Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train.

ConvLab is an open-source multi-domain end-to-end dialogue system platform offering the annotated MultiWOZ dataset and associated pre-trained reference models [61]. We reuse the rule-based dialogue state tracker from ConvLab to track the information that emerges during interactions between users and the dialogue agent. Besides, an agenda-based [115] user simulator is embedded and used for multi-domain dialogue scenarios.

Evaluation metrics Before a conversation starts, a user goal will be randomly sampled. The user goal consists of two parts: (1) the constraints on different domain slots or booking requirements, e.g., *Restaurant_Inform_Food* = *Thai*; (2) the slot values that show what the user is looking for, e.g., *Restaurant_Request_phone* = *?*. The task is completed successfully if a dialogue agent has provided all the requested information and made a booking according to the requirements. We use *average turn* and *success rate* to evaluate the efficiency and level of task completion of dialogue agents.

4.4.2 Architecture and training details

Variational autoencoder The encoder is a two-layer MLP that takes the discrete state representation (392 dimensions) as input and outputs two intermediate embeddings (64 dimensions) corresponding to the mean and the variance, respectively. For inference, we regard the mean μ as the embedded representation for a given state input s.

Auxiliary generator The auxiliary generator takes randomly sampled Gaussian noise as input and outputs a continuous state representation and a one-hot action embedding. The input noise is fed to a one-layer MLP first followed by the state generator Gen_s and action generator Gen_a. Gen_s is implemented with a two-layer MLP which output is the simulated state representation (64 dimensions) corresponding to the input noise. The main component of Gen_a is a two-layer MLP followed by a Gumbel-Softmax function. The output of the Gumbel-Softmax function is a one-hot representation (300 dimensions). Specifically, we implemented the "Straight-Through" Gumbel-Softmax Estimator [46] and the temperature for the function is set to 0.8.

Discriminator The discriminator is a three-layer MLP that takes as input the concatenation of latent state representation (64 dimensions) and one-hot encoding of the action (300 dimensions). During adversarial training, the real samples come from the real human dialogues in the training set while the simulated samples have three different sources. The main source is the output of the auxiliary generator introduced above. The second one is a random sample of state-action pairs from the training set where the action in each pair is replaced with a different one to build a simulated state-action pair. As a third source, we keep a history buffer with size $10k$ to record the simulated state-action pairs from the generator, where the state-action pairs are replaced randomly by the newly generated pairs from the generator. To strengthen the reward, we incorporate the human feedback r_{Human} into the pre-trained reward function. As the final reward function to train the dialogue agent we use the mixed reward $r_{GAN\text{-}VAE} = r_{Human} + \log(D(s, a))$.

4.4.3 Reinforcement learning methods

In this chapter, we validate our pre-trained reward using two different types of RL methods: Deep Q-learning (DQN) [92], which is an off-policy RL algorithm, and PPO [118], which is a policy-gradient-based RL method. To increase the training speed, we extend the vanilla DQN to WDQN, where the real dialogue state-action pairs from the training set are used to warm up the dialogue policy at the very beginning and then gradually removed from the training buffer. We implemented the DQN and PPO algorithms by utilizing the RL training modules in ConvLab.

4.4.4 Baselines

The handcrafted reward function r_{Human} is defined at the conversation level as follows: if the dialogue agent successfully accomplishes the task within T turns, it will receive $T * 2$ as a reward; otherwise, it will receive $-T$ as a penalty. T is the maximum number of dialogue turns. T is set 40 for experimentation. Furthermore, the dialogue agent will receive -1 as an intermediate reward during the dialogue to encourage shorter interactions.

In terms of DQN-based methods, we have *DQN*(*Human*) trained with r_{Human} and *DQN*(*GAN-VAE*) trained with $r_{GAN\text{-}VAE}$. We also develop a variant *DQN*(*GAN-AE*) by replacing the variational autoencoder in *DQN*(*GAN-VAE*) with an vanilla autoencoder. With respect to WDQN, we provide three different dialogue agents trained with reward functions from *Human*, *GAN-AE*, and *GAN-VAE*.

In terms of PPO-based methods, we implemented Generative Adversarial Imitation Learning (GAIL) [41] and Adversarial Inverse Reinforcement Learning (AIRL) [133]. In *GAIL*, the reward is provided with a discriminator where its parameter will be updated during the adversarial training process. *AIRL* is an adversarial learning method as well. The difference is that the discriminator in *GAIL* is replaced with a reward function that acts at the action level, which takes as input the dialogue state, system action, and the next state and returns the reward for the given dialogue action. For a fair comparison, both the *GAIL* discriminator and the *AIRL* reward model have been pre-trained. We also utilize teacher-forcing [7] for human dialogues to stabilize the adversarial training process.

Next, we report the average performance by running the same method 8 times with different random seeds.

4.5 Experimental Results

4.5.1 Results with DQN-based agents

Figure 4.2 plots the results of DQN-based methods with different reward functions but the same user simulator. The dialogue policy trained with *GAN-VAE* shows the best performance in terms of convergence speed and success rate. In comparison with *GAN-VAE* and *GAN-AE*, the updating signal from the handcrafted reward function r_{Human} can still guide the dialogue policy to a reasonable performance but with a slower speed. This suggests that denser reward signals could speed up dialogue policy training. Moreover,

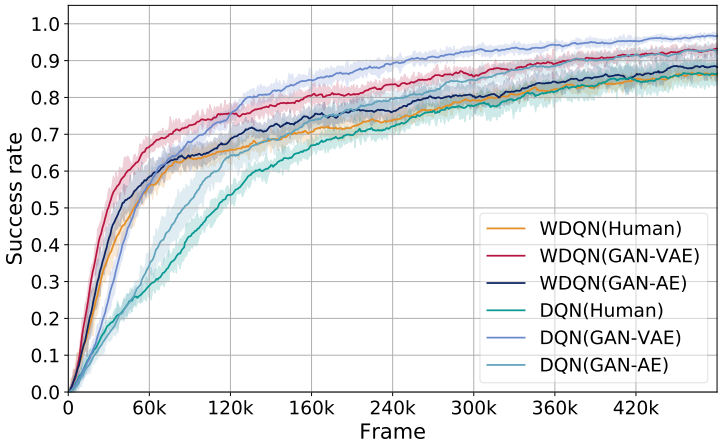


Figure 4.2: The learning process of DQN-based dialogue agents with different reward functions. One frame represents one interaction between the user and the dialogue agent.

the policy with r_{Human} converges to a lower success rate compare to *GAN-VAE* and *GAN-AE*. It suggests that, to some extent, the pre-trained reward functions have mastered the underlying information to measure the quality of given state-action pairs. The knowledge that the reward function learned during the adversarial learning step could be generalized to unseen dialogue states and actions to avoid a potential local optimum. In contrast, the dialogue agent *DQN*(*Human*) only relies on the final reward signal from the simulator at the end of the dialogue, which cannot provide enough guidance to the ongoing turns during conversations. This could be the reason why *DQN*(*Human*) shows lower success rate compare to *DQN*(*GAN-VAE*) and *DQN*(*GAN-AE*). The representation quality of the learned state embeddings leads to higher *GAN-VAE* performance over *GAN-AE*, because *VAE* generalizes better thereby bringing more benefits to the reward functions.

Examining closer WDQN agents, we can see all three methods achieve their inflection points after the first $30k$ frames. Comparing *DQN*(*Human*) and *WDQN*(*Human*), we found that the real human-human generated dialogue pairs from the training set do alleviate the problem of sparse reward provided by r_{Human} at the start stage of policy training. Similar results could be observed from agents trained with the pre-trained reward function $r_{GAN\text{-}VAE}$. After $24k$ frames, the *WDQN*(*Human*) curve coincides in position with *DQN*(*Human*) and they converge to the same point in the end. The faster convergence speed on *WDQN*(*Human*) did not bring a higher success rate because the dialogue policy still has no access to precise intermediate reward signals for the ongoing dialogue turns.

Table 4.1 reports the final performance of different dialogue agents during test time. All the agents have been trained with $500k$ frames and we save and evaluate the model that has the best performance during the training stage. Interestingly, *DQN*(*GAN-VAE*) outperforms *WDQN*(*GAN-VAE*) while *WDQN*(*Human*) beats *DQN*(*Human*). The

Table 4.1: The final performance of DQN-based dialogue agents with different reward functions.

Dialogue agent	Success Rate	Average Turn
WDQN$_{keep}$(Human)	0.741	19.144
WDQN$_{keep}$(GAN-AE)	0.879	15.118
WDQN(Human)	0.906	13.580
WDQN(GAN-AE)	0.911	13.298
WDQN(GAN-VAE)	0.937	12.260
DQN(Human)	0.870	14.960
DQN(GAN-AE)	0.953	12.300
DQN(GAN-VAE)	**0.985**	**11.040**

warming-up stage in *WDQN*(*GAN-VAE*) does improve the training speed but it results in a lower final success rate. The potential reason is that the real human-human dialogue can bring a strong update signal at the beginning of the training process but at the same time limits the exploration ability of the agent. To verify this finding, we designed two more WDQN agents: *WDQN*$_{keep}$(*Human*) and *WDQN*$_{keep}$(*GAN-AE*), which keep expert dialogues examples during the entire training phase, rather than removing them gradually. Their performance is shown in Table 4.1. As to agents trained with r_{Human}, there is a huge performance gap, *WDQN*(*Human*) outperforms *WDQN*$_{keep}$(*Human*) almost by 15%. The difference in the performance of *WDQN*$_{keep}$(*GAN-AE*) and *WDQN*(*GAN-AE*) is significantly smaller because the pre-trained reward function brings more precise and consistent update signals that are explored and disclosed during the adversarial training step.

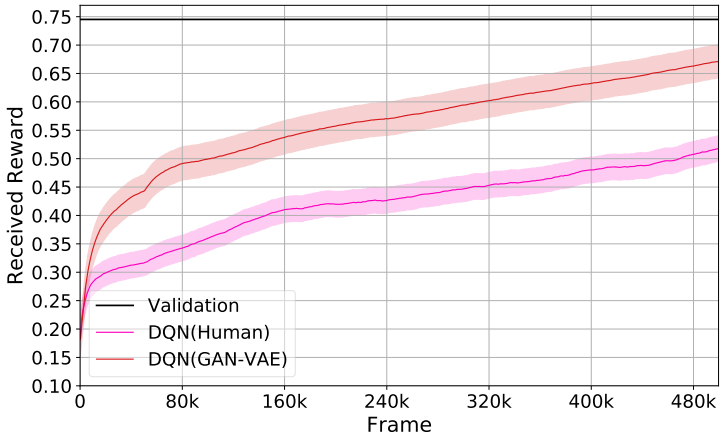


Figure 4.3: The reward returned by the pre-trained reward function during dialogue policy training.

Figure 4.3 shows curves presenting the reward changes during the RL training. The

curve *Validation* denotes the average reward received based on the real human-human dialogues, which can be regarded as the human performance evaluated by the pre-train reward function $r_{GAN\text{-}VAE}$ and it is ~ 0.74.[2] For *DQN*(*Human*) and *DQN*(*GAN-VAE*) training, we feed generated in real-time dialogue batches to reward function $r_{GAN\text{-}VAE}$. We can see that both approaches are getting a high reward, but *DQN*(*GAN-VAE*) is growing faster because $r_{GAN\text{-}VAE}$ is used for the training of *DQN*(*GAN-VAE*). That is a promising finding since we can suggest that a well-trained reward function can be utilized not only to guide the dialogue policy training but also to judge the quality of different agents.

4.5.2 Results with PPO-based agents

As for *GAIL* and *AIRL*, the reward functions are updated on the fly, and therefore we can only employ policy gradient-based RL algorithms. We use PPO algorithms to train the dialogue agent with different reward functions. Before initiating training, we first warm-up all the dialogue agents with human dialogues via imitation learning. As a result, the warmed-up agents share similar success rates which are $\sim 33\%$. We also pre-train discriminators in *GAIL* and reward models in *AIRL* utilizing positive examples from the training set and negative examples from the pre-trained dialogue agents.

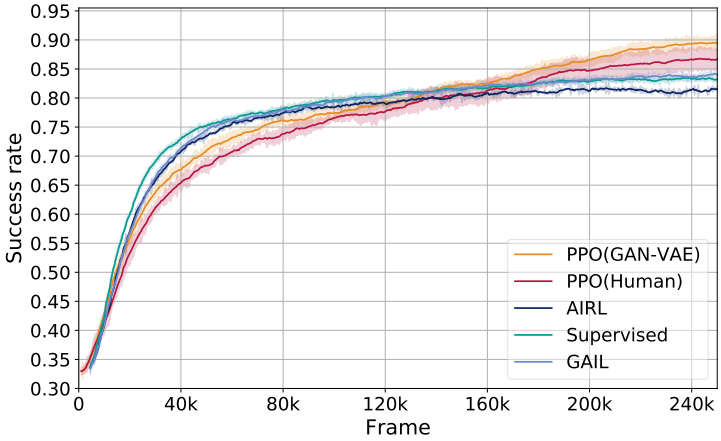


Figure 4.4: The learning process of PPO-based dialogue agents with different reward functions.

Figure 4.4 demonstrates that in terms of success rate *GAIL* and *AIRL* rise faster than *PPO(GAN-VAE)* and *PPO(Human)* during first $120k$ frames. Then both methods flattened and converged at $\sim 81\%$. It is important to note, that we utilize teacher-forcing in the adversarial step by feeding human-human dialogues to the agents every several frames while training *GAIL* and *AIRL*. Due to the large task action space, it is nearly impossible to successfully train a high-quality dialogue agent without teaching-forcing

[2] Ideally, the reward on human dialogues should be equals to 0.5 because the discriminator is not able to distinguish the simulated dialogues from real human-human ones after generator and discriminator converge according to Eq. 4.7.

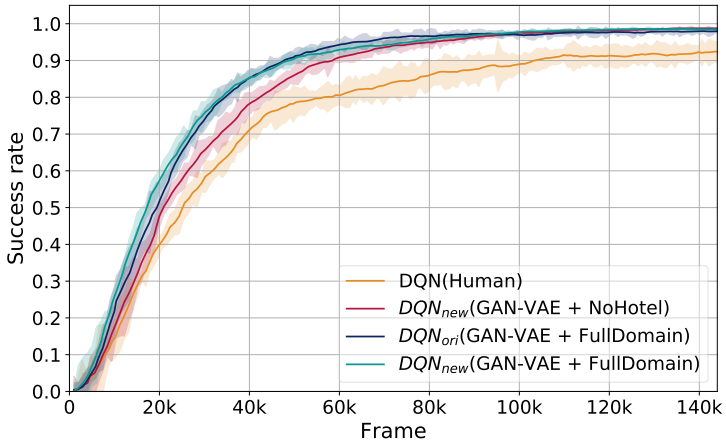


Figure 4.5: The learning process of dialogue agents in different domains.

steps in adversarial learning methods. The agent called *supervised* represents the setup where we discard the training signals from the discriminators or the reward models in *GAIL* and *AIRL* and only train the policy network using teacher-forcing with the same frequency. We can observe that the adversarial training signal in *GAIL* and *AIRL* degenerates the performance of supervised learning methods.

Discussion

We explored various parameters for *GAIL* and *AIRL* setups, unfortunately unsuccessful. The potential reason is ConvLab has 300 actions, and it is intractable for a dialogue agent to explore the action space relying only on the sparse positive reward signals which can easily lead to a local optimum. Takanobu et al. [133] successfully applied *AIRL* to learn dialogue policy, but the considered size of action space was only half compared to our setup. More importantly, Takanobu et al. [133] formulated dialogue policy learning as a multi-label classification task where it is easier to achieve a higher success rate by selecting as many actions as possible in one turn. Moreover, DQN-based RL algorithms are not applicable in their setup. In comparison, our agent *PPO(GAN-VAE)* can achieve higher performance in the more commonly used setup. Comparing *PPO(GAN-VAE)* and *PPO(Human)*, we can verify our claim that the dialogue agent benefits from the pre-trained reward function $r_{GAN\text{-}VAE}$. As shown in Figure 4.2 and Figure 4.4, the agents trained using the hand-crafted reward function, such as *DQN*(*Human*) and *PPO(Human)*, share a similar final performance $\sim 87\%$. Another important finding the DQN-based agents benefit more compared to the PPO-based ones from incorporating the reward signals from the same reward function $r_{GAN\text{-}VAE}$.

4.5.3 Transfer learning with a pre-trained reward function

To define the action space, we utilize the 300 most frequent actions from the MultiWoz dataset and use one-hot embedding to represent them. As shown in Figure 4.1, the

action, and the state representations are concatenated to form a specific state-action pair. This approach ignores the relations between different actions. For example, *Restaurant_Inform_Price* and *Restaurant_Request_People* should be close for the same conversation since they happen to be in the same domain. However, even for different domains, connections between actions are possible, e.g., *Inform_Price* and *Request_People* can also happen in the *Hotel* domain, corresponding to actions *Hotel_Inform_Price* and *Hotel_Request_People*. We ask ourselves if we can transfer the knowledge learned in existing domains to a new domain, which we have never seen before via the pre-trained reward function. To answer this question, we first reformulate the action representation as a concatenation of three different segments: *Onehot*(*Domain*), *Onehot*(*Diact*), *Onehot*(*Slot*). Following this approach, actions containing similar information will be linked through the corresponding segments in their representation. Utilizing this formulation, we retrained our reward function in selected domains and incorporate it into the training of a dialogue agent in a new unseen domain. Concretely, we train the reward function based on the following domains: *Restaurant, Bus, Attraction, and Train*. As a testing domain, we pick *Hotel* since it has the most slot types and some of them are unique, such as *Internet, Parking, Stars*. DQN_{ori} in Figure 4.5 corresponds to the dialogue agent trained with all domains and the action is represented with a single one-hot embedding. By replacing the action representation in DQN_{ori} with the new action formulation we get agent – DQN_{new}. Based on the obtained results, we can conclude DQN_{new}(*GAN-VAE + NoHotel*) benefits from the reward function trained in different domains and it outperforms *DQN*(*Human*). As expected, the agents DQN_{new}(*GAN-VAE + FullDomain*) and DQN_{ori}(*GAN-VAE + FullDomain*), which are trained using reward from all domains, have a better performance compared to DQN_{new}(*GAN-VAE + NoHotel*).

4.6 Conclusion

In this chapter, we have proposed a guided dialogue policy training method without using adversarial training in the loop. First, we trained the reward model with an auxiliary generator. Then the trained reward model was incorporated into a common reinforcement learning method to guide the training of a high-quality dialogue agent. By conducting extensive experimentation, we demonstrated that the proposed methods achieve remarkable performance, in terms of task success, as well as the potential to transfer knowledge from previously utilized task domains to new ones.

With respect to the answer to research question RQ3 it is obvious that off-policy methods can benefit from data-driven reward functions in task-oriented dialogue systems. In this work, we only use it to train dialogue agents and additionally regard it as a bridge to transfer knowledge among domains. In the future, we can dig deeper into the usage of the recovered reward functions. For example, it is quite challenging to evaluate the quality of generated responses in open-domain dialogue systems because one dialogue context could have multiple responses and all of them are reasonable and appropriate. It is promising if we can train a data-driven reward function with huge amounts of dialogue collections (e.g., training data-driven reward functions with a pre-trained language model, like BERT [20]) and then use the trained reward function as a tool to judge the quality of generated responses.

In this chapter, to transfer the discrete state space to continuous space, we rely on variational autoencoders and this setup may lead to the mode collapse in the training of variational autoencoders. Besides, we need high-quality human-human dialogue collections to train the reward function and reliable user simulators to train the dialogue agent with RL methods. With more uncontrollable factors involved in the whole pipeline, the errors produced in the previous modules may be accumulated and lead to undesirable results in the dialogue agent step. These potential issues push us to rethink the use of RL methods in task-oriented dialogue systems and revisit some traditional dialogue training methods in the following chapter.

5

Rethinking Supervised Learning and Reinforcement Learning in Task-Oriented Dialogue Systems

This chapter is aimed at answering the following research question:

> *Are we really making progress in applying only Reinforcement Learning (RL) to dialogue policy learning for Task-oriented dialogue systems (TDSs)?*

We propose two supervised learning approaches and one adversarial learning method to train the dialogue policy for Task-oriented dialogue systems without building user simulators. We show the advantages and disadvantages of different dialogue policy learning methods.

5.1 Introduction

The aim of dialogue policies in TDSs is to select appropriate actions at each time step according to the current context of the conversation and user feedback [13]. In early work, dialogue policies were manually designed as a set of rules that map the dialogue context to a corresponding system action [142]. The ability of rule-based solutions is limited by the domain complexity and task scalability. Moreover, the design and maintenance of these rules require a lot of effort and domain knowledge.

Due to recent advantages in deep learning and the availability of labeled conversational datasets, *supervised learning* can be employed for dialogue policy training to overcome the disadvantages of rule-based systems. The downside of the supervised learning approach is that the dialogues observed in the datasets are unlikely to represent all possible conversation scenarios; in some extreme cases, the required conversational dataset cannot be collected or acquiring it might be cost-prohibitive.

The success of RL in other areas holds promises for dialogue Policy Learning (PL) [147]. Using RL techniques, we can train dialogue policies and optimize automatically, from scratch and utilizing interactions with users [34, 130]. In RL-based solutions, the dialogue system takes actions that are controlled by the dialogue policy,

This chapter was published as [77].

and user feedback (the *reward signal*), which is provided when the dialogue is finished, is utilized to adjust the initial policy [21, 101, 149]. In practice, reward signals are not always available and may be inconsistent [129]. As it is not practical to ask for explicit user feedback for each dialogue during policy training, different strategies have been proposed to design a rule-based user simulator along with a reward function that can approximate the real *reward function* which exists only in each user's mind. Designing an appropriate user simulator and accurate reward function requires strong domain knowledge. This process has the same disadvantages as rule-based dialogue systems [138]. The difference is that rule-based approaches to system design meet this problem at the dialogue agent side while rule-based user simulators need to solve it at the environment side.

If the task is simple and easy to solve, why not just build a rule-based system rather than a user-simulator that is then used with RL techniques to train the dialogue system, where more uncontrollable factors are involved? And if the task domain is complex and hard to solve, is it easier to design and maintain a complicated rule-based user simulator than to build a rule-based dialogue agent? Supervised learning methods do not suffer from these issues but require labeled conversational data; in some exceptional cases, if the data cannot be collected for privacy reasons, RL is the solution. However, collecting labeled data is feasible for many applications [11, 143, 148]. Therefore, in this chapter we seek to answer the following research question: *Are we really making progress in TDSs focusing purely on advancing RL-based methods?*

To address this question, we introduce three dialogue PL methods that do not require a user simulator. The proposed methods can achieve comparable or even higher performance compared to state-of-the-art (SOTA) RL methods. The first method utilizes an action decoder to predict dialogue combinations. The second method regards the dialogue PL task as a multi-label classification problem. Unlike previous work, we assign a dense layer to each action label in the action space. Based on the second method, we propose an adversarial learning method for dialogue PL without utilizing RL. To backpropagate the loss from the reward model to the policy model, we utilize the Gumbel-Softmax to connect the policy model and the reward model in our third method. We compare our methods with RL and adversarial RL based dialogue training solutions to show how we can achieve comparable performance without utilizing costly user simulator.

To summarize, our contributions are:

- A dialogue action decoder to learn the dialogue policy with supervised learning.
- A multi-label classification solution to learn the dialogue policy.
- A simulation-free adversarial learning method to improve the performance of dialogue agents.
- Achieving SOTA performance in dialogue PL with fewer efforts and costs compared to existing RL-based solutions.

5.2 Related Work

A number of RL methods, including deep Q-learning [71, 78, 82, 99, 131] and policy gradient methods [21, 149], have been applied to optimize dialogue policies by interact-

ing with real users or user simulators. RL methods help the dialogue agent to explore contexts that may not exist in previously observed data. A key component in RL is the quality of the reward signal used to update the agent policy. Most existing RL-based methods require access to a reward signal based on user feedback or a pre-defined one if the feedback loop is not possible. Besides, designing a good reward function and a realistic user simulator is not easy as it typically requires strong domain knowledge, which is similar to the problem that rule-base methods meet. Peng et al. [100] propose to utilize adversarial loss as an extra critic in addition to the main reward function based on task completion. Inspired by the success of adversarial training in other NLP tasks, Liu and Lane [83] propose to learn dialogue rewards directly from dialogue samples, where a dialogue agent and a dialogue discriminator are trained jointly. Following the success of inverse reinforcement learning (IRL) in different domains, Takanobu et al. [133] employ *adversarial IRL* to train the dialogue agent. They replace the discriminator in GAIL [41] with a reward function with a specific architecture. The learned reward function can provide a stable reward signal and adversarial training can benefit from high-quality feedback.

Compared to existing RL based methods, we propose a strategy that can eliminate designing a user simulator and sensitive parameter-tuning process while bringing a significant performance improvement with respect to a number of metrics. The absence of user simulators involved will largely reduce the required domain knowledge and supervised learning can lead to robust agent performance.

5.3 Multi-Domain Dialogue Agent

dialogue state tracking (DST) In a standard Task-oriented dialogue system (TDS) pipeline [61, 70], the rule-based DST is deployed to keep track of information emerging in interactions between users and the dialogue agent. The output from the natural language understanding (NLU) module is fed to the DST to extract information, including informable slots about the constraints from users and requestable slots that indicate what users inquire about. In our setup, the dialogue agents and user-simulators are interacting through predefined dialogue actions therefore no NLU is involved. Besides, a belief vector is maintained and updated for each slot in every domain.

Dialogue state We formulate a structured state representation s_t according to the information resulting from the DST at time step t. There are 4 main types of information in the final representation: (1) corresponding to the embedded results of returned entities for a query, (2) the last user action, (3) the last system action, and (4) the belief state from the rule-based state tracker. The final state representation s is a vector of 553 bits.

Dialogue action We regard the dialogue response problem as a multi-label prediction task, where in the same dialogue turn, several atomic dialogue actions can be covered and combined at the same moment. In the action space, each action is a concatenation of domain name, action type and slot name, e.g. *'attraction-inform-address'*, which we call an *atomic action*[1]. Lee et al. [61] propose that the action space covers both

[1]There are 166 atomic actions in total in the action space.

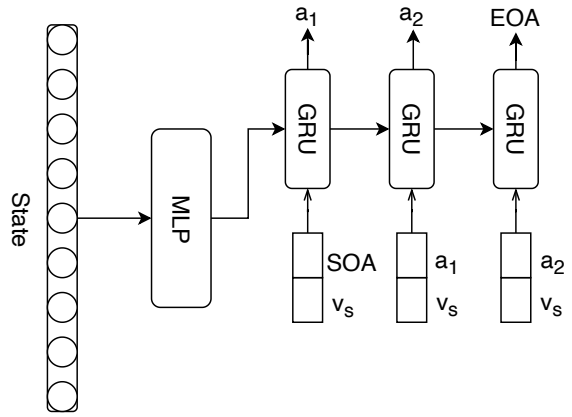


Figure 5.1: Architecture to approximate a dialogue policy with an action decoder. *SOA* and *EOA* are special actions corresponding to the starting signal and ending signal respectively.

the atomic action space and the top-k most frequent atomic action combinations in the dataset and then the dialogue PL task can be regarded as a single label classification task. However, the expressive power of the dialogue agent is limited and it is beneficial if the agent can learn the action structure from the data and this could lead to more flexible and powerful system responses.

5.4 Dialogue Policy Learning (PL)

5.4.1 PL as a sequential decision process

Different atomic dialogue actions contained in the same response are usually related to each other. To fully make use of information contained in co-occurrence dependencies, we decompose the multi-label classification task in dialogue PL as follows. Assuming the system response consists of two atomic actions, *'hotel-inform-address'* and *'hotel-inform-phone'*, the model takes the dialogue state as input and predict the atomic actions sequentially. The path could be described as either *'hotel-inform-address'* $\rightarrow$ *'hotel-inform-phone'* or *'hotel-inform-phone'* $\rightarrow$ *'hotel-inform-address'*. Before the training stage, the relative order of all the atomic actions will be predefined and fixed. Following this solution, we apply a GRU-based [15] decoder to model the conditional dependency between the actions in one single turn as shown in Figure 5.1.

The proposed model first extracts state features v_s by feeding the raw state input s to an Multilayer Perceptron (MLP). In the next state, the state representation v_s will be used as the initial hidden state h_0 of action decoder GRU. To avoid information loss during decoding, the input to the action decoder is:

$$input_t = embedding(a_{t-1}) \oplus v_s. \tag{5.1}$$

The starting input $input_0$ is the concatenation of starting action *SOA* and state represen-

tation v_s. a_{t-1} denotes the dialogue action in the prediction path at time step $t-1$ and $embedding(a)$ returns the action embedding of the given action a. In the next steps, actions will be generated consecutively according to:

$$o_t, h_t = GRU(input_t, h_{t-1}), \tag{5.2}$$

where o_t is the output of the action decoder. We use cross-entropy to train the action decoder together with the MLP for feature extraction. We use beam-search to find the most appropriate action path.

5.4.2 PL with adversarial learning

Next, we introduce an adversarial learning solution, *DiaAdv*, to train the dialogue policy without a user simulator along with a dialogue discriminator. Feedback from the discriminator is used as a reward signal to push the policy model to interact with users in a way that is indistinguishable from how a human agent completes the task. However, since the output of the dialogue policy is a set of discrete dialogue actions, it is difficult to pass the gradient update from the discriminator to the policy model. To cross this barrier, we propose to utilize the Gumbel-Softmax function [46] to link the discriminator to the generator. Next, we will give a brief introduction about the dialogue policy model and the dialogue discriminator. Afterward, we will show how we can utilize Gumbel-Softmax to backpropagate the gradient.

Dialogue policy To generate dialogue actions, we employ an MLP as the action generator Gen_{sa} followed by a set of Gumbel-Softmax functions, where each function corresponds to a specific action in the atomic action space (Figure 5.2) and the output of each function has two dimensions. We first introduce how it works when there is only one Gumbel-Softmax function in the setting and then extend it to multiple functions. The Gumbel-Max trick [38] is commonly used to draw samples u from a categorical distribution with class probabilities p. The process of Gen_θ can be formulated as follows:

$$p = \text{MLP}(s) \tag{5.3}$$

$$u = one\_hot(\arg\max_i [g_i + \log p_i]), \tag{5.4}$$

where g_i is independently sampled from Gumbel (0, 1). However, the $\arg\max$ operation is not differentiable, thus no gradient can be backpropagated through u. Instead, we can employ the soft-argmax approximation [46] as a continuous and differentiable approximation to $\arg\max$ and to generate k-dimensional sample vectors below:

$$y_i = \frac{\exp((\log(p_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(p_j) + g_j)/\tau)}, \tag{5.5}$$

for $i = 1, \ldots, k$. In practice, τ should be selected to balance the approximation bias and the magnitude of gradient variance. In our case, p corresponds to the dialogue action status distribution $p(a_l^i|s)$ where $l \in \{0, \ldots, k-1\}$ and $i \in \{1, \ldots, m\}$. In our setting,

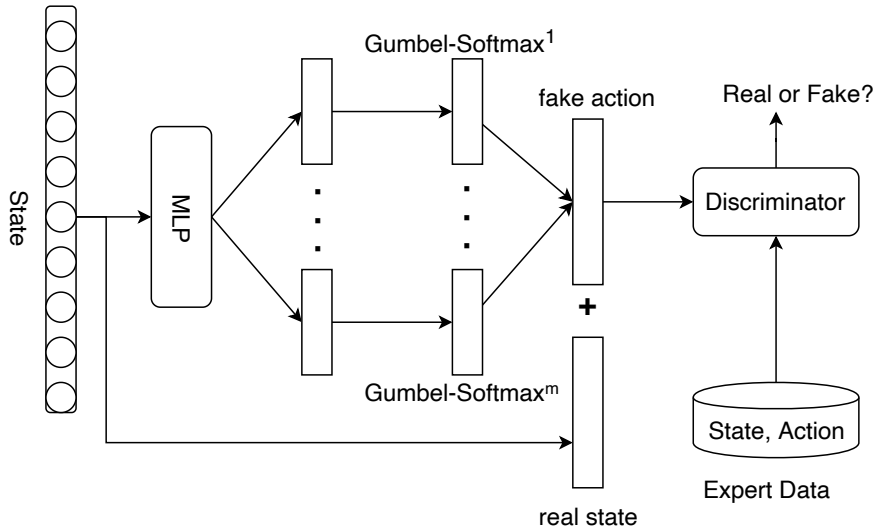


Figure 5.2: Architecture to approximate the dialogue policy with adversarial learning. The dialogue policy dialogue discriminator is linked to the dialogue policy through a set of Gumbel-Softmax functions; + denotes the concatenation operation.

k is set to 2 and each dimension denotes one specific action status, which could be 1 if selected or 0 if not selected. m is set to the size of the action space – 166. By taking into account the multiple actions, we rewrite the sampled vector y as y_l^i where l and i denote the corresponding dialogue action status and the i_{th} atomic action in the action space respectively. The final combined action is[2]:

$$a_{fake} = y_0^1 \oplus y_1^1 \oplus \ldots \oplus y_0^{166} \oplus y_1^{166}. \tag{5.6}$$

Next, the generated action a_{fake} is fed to the reward model D_ω along with the corresponding state s. The dialogue policy Gen_θ aims to get a higher reward signal from the discriminator D; the training loss function for the generator Gen_θ is:

$$L_G(\theta) = -\mathbb{E}_{s,a_{fake} \sim Gen}(D_\omega(s, a_{fake})). \tag{5.7}$$

Dialogue reward As to the dialogue discriminator, we build a reward model D_ω that takes as input the state-action pair (s, a) and outputs the reward $D(s, a)$. Instead of using a discriminator to predict the probability of a generated state-action pair as being real or fake, inspired by Wasserstein GANs [4], we replace the discriminator model with a reward model that scores a given pair (s, a). Since the reward model assigns a higher reward to the real data and a lower value to the fake data, the objective can be given as the average reward it assigns to the correct classification. Given an equal mixture of real data samples and generated samples from the dialogue policy Gen_θ, the

[2] $Dim(a_{fake}) = 166 * 2$.

loss function for the reward model D_ω is:

$$L_D(\omega) = -\mathbb{E}_{s,a_{fake}\sim Gen_\theta}(D_\omega(s, a_{fake})) \tag{5.8}$$
$$+\mathbb{E}_{s,a\sim data}(D_\omega(s, a))). \tag{5.9}$$

During training, the policy network and the reward model are be updated in an alternating manner.

5.4.3 PL as multi-label classification with dense layers

We introduced *DiaAdv*, which can bridge the policy network and the reward model together utilizing Gumbel-Softmax functions. A by-product of this framework is the policy network with dense layers and a set of Gumbel-Softmax functions. If we discard the Gumbel-Softmax functions but keep the dense layers, we obtain a new model, *DiaMultiDense*, to solve the multi-label classification problem. Each dense layer corresponds to a specific dialogue action and the output of the dense layer has two dimensions denoting the two possible values for action status, *selected* and *not selected*. We expect the dense layers can extract informative information particularly for their corresponding actions and discard noisy information. During inference, the two possible values for the status of action will be compared and the higher one will be the label for the current dialogue action. *DiaMultiDense* can be regarded as a simple but efficient state de-noising method for dialogue PL with multi-label classification.

5.5 Experimental Setup

MultiWOZ dataset is a multi-domain dialogue dataset with 7 distinct domains[3], and $10,438$ dialogues [11]. The main used scenario is a dialogue agent is trying to satisfy the tourists' demands such as booking a restaurant or recommending a hotel with specific requirements. Each dialogue trajectory is decomposed into a set of state-action pairs with the same TDS that is used for training. In total, we have $56,700$ dialogue state-action pairs in the training set, with $7,300$ in the validation set, and $7,300$ in the test set.

Baselines Three types of baseline are considered:

(B1): *Supervised Learning*, where the dialogue action selection task is regarded as a multi-label classification problem.

(B2): *Reinforcement Learning*, where the reward function is handcrafted and defined as follows: at the end of a dialogue, if the dialogue agent accomplishes the task within T turns, it will receive $T * 2$ as a reward; otherwise, it will receive $-T$ as a penalty. T is the maximum number of turns in each dialogue; we set it to 40 in all experiments. Furthermore, the dialogue agent will receive -1 as an intermediate reward during the dialogue to encourage shorter interactions. In our experiments, we used three methods, namely: *GP-MBCM* [35], *ACER* [140], *PPO* [118].

[3] Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train.

(B3): *Adversarial learning*, where the dialogue agent is trained with a user simulator, we conduct comparisons with two methods: *GAIL* [41] and *GDPL* [133]. The dialogue agents in *GAIL* and *GDPL* are both PPO agents while these two methods have different reward models. We report the performance of *ALDM* [83] for completeness.

5.5.1 Training setup

DiaSeq With respect to *DiaSeq*, we use a two-layer MLP to extract features from the raw state representation. First, we sort the action order according to the action frequency in the training set. All action combinations in the dataset will be transferred to an action path based on the action order. Three special actions – *PAD*, *SOA*, *EOA*, corresponding to padding, start of action decoding and end of action decoding – are added to the action space for action decoder training. We use beam search to predict the action combinations and the beam size is set to 6. The action embedding size is set to 30; the hidden size of the GRU is 50.

DiaAdv For the policy network of *DiaAdv*, a two-layer MLP is used to extract state features followed by 166 dense layers and Gumbel-Softmax functions consecutively. To sample a discrete action representation, we implemented the "Straight-Through" Gumbel-Softmax Estimator [46]; the temperature τ for each function is set to 0.005. As to the discriminator, a three-layer MLP takes as input the concatenation of dialogue state and action and outputs a real value as the reward for the state-action pair.

DiaMultiDense We reuse the policy network from *DiaAdv* except for the Gumbel-Softmax functions.

GDPL The policy network and value network are three-layer MLPs.

PPO The policy network in PPO shares the same architecture as *GDPL*. The difference is that the reward model is replaced with a handcrafted one.

GAIL *GAIL* shares the same policy network as *GDPL*. The discriminator is a two-layer MLP taking as input the state-action pair.

DiaMultiClass The policy network is a three-layer MLP and trained with cross-entropy. It has the same architecture as the policy network in *GDPL*.

We reuse the reported performance of GP-MBCM, ACER, and ALDM from [133] since we share the same TDS and user simulator. The methods based on RL or adversarial learning are pre-trained with real human dialogues.

5.5.2 Evaluation metrics

Before a conversation starts, a user goal will be randomly sampled. The sampled user goal mainly contains two parts of information. The first part is about the constraints of different domain slots or booking requirements, e.g. *'restaurant-inform-food' = 'Thai'*,

Table 5.1: The performance of different dialogue agents, which is calculated based on the average results by running each method 5 times. * indicates statistically significant improvements ($p < 0.005$) using a paired t-test over the *GDPL* success rate and the proposed methods.

Dialogue agent	Turn	Match	Rec	F1	Success rate
GP-MBCM	2.99	0.44	–	0.19	28.9
ACER	10.49	0.62	–	0.78	50.8
PPO (human)	15.56	0.60	0.72	0.77	57.4
ALDM	12.47	0.69	–	0.81	61.2
GDPL	7.80	0.81	0.89	0.87	81.7
GAIL	7.96	0.81	0.87	0.86	80.5
DiaMultiClass	12.66	0.58	0.71	0.79	57.2
DiaMultiDense	9.33	0.85	0.94	0.87	86.3*
DiaSeq	9.03	0.81	0.88	0.85	81.6
DiaAdv	8.80	0.85	0.94	0.85	87.4*

'restaurant-infor-area' = 'east', *'restaurant-book-people' = 4* which means the user wants to book a table for 4 persons to have Thai food in the east area. The information contained in the second part is about the slot values that the user is looking for, such as *restaurant-request-phone = ?*, *'restaurant-request-address' = ?*, which means the user wants to know the phone and address of the recommended restaurant. We use *Match*, *Recall*, *F1 score* to check if all the slot constraints and requested slot information have been satisfied. *F1 score* evaluates whether all the requested information has been provided while *Match* evaluates whether the booked entities match the indicated constraints. We use *Average Turn* and *Success rate* to evaluate the efficiency and level of task completion of dialogue agents. If an agent has provided all the requested information and made a booking according to the requirements, the agent completes the task successfully.

5.6 Results and Discussion

5.6.1 Performance of different dialogue agents

Table 5.1 shows the performance of different dialogue agents. With respect to the success rate, *DiaAdv* manages to achieve the highest performance by 6% compared to the second-highest method *GDPL*. However, *DiaAdv* is not able to beat *GDPL* in terms of average turns. A possible reason is that *GDPL* can generate more informative and denser dialogue action combinations. With a user simulator in the training loop, the dialogue agent can explore more unseen dialogue states in the dataset. Furthermore, the same user simulator will be used to test the dialogue agent and the dialogue agent will benefit from what he has explored in the training stage. However, more informative and denser responses will not guarantee all the users' requirements will be satisfied and this will lead to a lower *Match* score as shown in Table 5.1.

Table 5.2: Total number of parameters for supervised learning models.

Dialogue agent	DiaSeq	DiaMultiClass	DiaMultiDense
#Parameters	251,000	184,000	133,000

Table 5.3: The performance of different dialogue agents with different numbers of expert dialogues. We only report *Average Turn* and *Success rate* here due to limited space.

Agent \ Dataset	MultiWOZ (0.1)		MultiWOZ (0.4)		MultiWOZ (0.7)	
	Turn	Success rate	Turn	Success rate	Turn	Success rate
DiaMultiClass	17.14	31.7	12.56	59.0	13.10	53.6
DiaSeq	10.77	70.4	9.99	75.5	9.35	77.2
DiaMultiDense	18.36	27.0	10.76	79.4	10.02	85.1
GDPL	9.21	21.2	8.49	68.0	8.10	73.3
DiaAdv	16.80	37.2	9.90	81.6	9.30	87.0

As to *DiaSeq*, it can achieve almost the same performance as *GDPL* from different perspectives while *GDPL* has a slightly higher *F1* score. However, the potential cost benefits of *DiaSeq* are huge since it does not require a user simulator in the training loop. The training of *DiaSeq* is well-understood and we can get rid of tuning the sensitive parameters in RL and Adversarial Learning. To sum up, *DiaSeq* is far more cost-efficient solution.

Another supervised learning method, *DiaMultiDense* achieves remarkable performance with respect to different metrics. Compared to the traditional solution *DiaMultiClass*, joining of dense layers as in *DiaMultiDense* brings a huge performance gain; it manages to beat *DiaMultiClass* on all the metrics. And it achieves higher *F1* score than *DiaAdv*. Since the only difference between *DiaMultiDense* and *DiaMultiClass* is that we replace the last layer of *DiaMultiClass* with a stack of dense layers, the change in the number of parameters may lead to the performance gap. We report the number of parameters of three supervised learning methods in Table 5.2. *DiaMultiDense* achieves the highest performance among these three methods while using the fewest parameters. We believe the dense layers have been trained to filter noisy information from the previous module and the final classification can benefit from the high-quality information flow.

5.6.2 User experience evaluation

Automatic metrics can only capture part of the performance difference between different dialogue agents. For example, we use the success rate to reflect the level of task completion and use turn numbers to represent the efficiency of dialogue agents. However, the final goal of a TDS is to assist real users to complete tasks. To fully evaluate system performance while interacting with real users, we launch an evaluation task on Amazon MTurk[4] to rate the user experience with the proposed dialogue systems.

[4]Amazon MTurk: `https://www.mturk.com/`

Table 5.4: Human evaluation results.

Dialogue pair	Win	Loose	Tie
DiaMultiDense vs. GDPL	42	50	8
DiaSeq vs. GDPL	50	44	6
DiaAdv vs. GDPL	39	51	10

For each evaluation task, we will first present an MTurk worker with a randomly sampled user goal, which contains the constraints about specific domain slots and some slot information that the user is looking for. In the next step, according to the sampled goal, two generated dialogues from two different dialogue agents are shown to the worker. The worker needs to pick the dialogue agent that provides a better user experience. Different factors will be taken into account, such as response quality, response naturalness, how similar it is compared to a real human assistant. If the worker thinks two dialogue agents perform equally good/bad or it's hard to distinguish which one is better, the option 'Neutral' can be selected. Four dialogue agents are evaluated: *GDPL*, *DiaSeq*, *DiaMultiDense* and *DiaAdv*, and there are three comparison pairs *DiaMultiDense-GDPL*, *DiaSeq-GDPL*, *DiaAdv-GDPL* since *GDPL* is regarded as the SOTA method. Each comparison pair has 100 dialogue goals sampled and 200 corresponding dialogues from two different dialogue agents. All the dialogue actions in the dialogue turns are translated into human-readable utterances with the language generation module from ConvLab [61]. Each dialogue pair is annotated by three MTurk workers. The final results are shown in Table 5.4.

The method *DiaAdv* can be regarded as an extension of *DiaMultiDense* by adding a classifier to provide a stronger training signal. According to the results from Section 5.6.1, these two methods do improve the success rate of dialogue agents. However, as shown in Table 5.4, while the success rate improves, the user experience degrades. According to Table 5.1, *GDPL* and *DiaAdv* have similar *F1* scores but the *DiaAdv* has a higher *Recall* value; this means that *DiaAdv* achieves a lower *Precision*. The unnecessary information mixed in the system response annoys users and results in lower user experience. Given the relatively large difference in terms of success rate, the trade-off between success rate and user experience should be carefully examined. From another perspective, it is understandable that *GDPL* can provide a better user experience because a pre-designed user simulator is involved and the discriminator will encounter more diverse state-action combinations that are not seen in the training data. In contrast, the discriminator in *DiaAdv* only has access to the training data and this limits its judging ability. This does not imply that having a user simulator in the loop is essential to provide high-quality user experience: *DiaSeq*, which is a completely supervised learning method, outperforms *GDPL*.

5.6.3 Discussion

How many expert dialogues are enough to train a dialogue agent with supervised learning? One motivation for dropping supervised learning and employing RL methods in TDSs is that building high-quality conversational datasets is expensive and

time-consuming. In contrast, training dialogue agents with a user-simulator is cheaper and more affordable in many cases. Since we have no control over how much domain knowledge should be involved to build a user-simulator, we are not able to measure the expense of a reliable user-simulator. However, we can conduct an experiment to show how many real human dialogues are required to train a high-quality dialogue agent.

Based on the original MultiWoZ dataset, we build three smaller subsets: MultiWoZ(0.1), MultiWoZ(0.4), MultiWoZ(0.7) by only keeping 10%, 40%, and 70% dialogue pairs from the original dataset, respectively. We retrain *DiaMultiClass*, *GDPL*, *DiaAdv*, *DiaMultiDense*, *DiaSeq* and report the performance in Table 5.3. With respect to supervised learning agents, with only 10% expert dialogue pairs, *DiaMultiClass* gets half the success rate compared to the original performance (Table 5.1). By adding 30% more dialogue pairs to the training set, *DiaMultiClass* can achieve the same performance 59% with the original success rate of 57.2%. Beyond this, *DiaMultiClass* does not benefit from the increase in expert dialogues and starts to fluctuate between 55% and 59%. In contrast, *DiaSeq* can achieve higher performance when there are only 10% expert dialogue pairs, and the success rate increases with the number of available expert dialogues. *DiaMultiDense* achieves the best performance with the same amount of expert dialogues compared to the other two supervised learning methods. The performance difference among the three supervised learning methods shows that the method itself is the main factor to influence the performance rather than the number of available expert dialogues in the given dialogue environment. To some extent, traditional *DiaMultiClass* does not exert the potential of a given dataset to the fullest in dialogue PL.

Can adversarial learning eliminate expert dialogues? As can be concluded from Table 5.3, *GDPL* and *DiaAdv* manage to improve the performance with the increasing number of expert dialogues. *GDPL* and *DiaAdv* have the reward models that are supposed to distinguish real dialogue pairs from the machine-generated ones. By observing more expert dialogues, the reward model can provide a dialogue policy with more reliable and consistent updating signals. Figure 5.3 shows the success rate gain

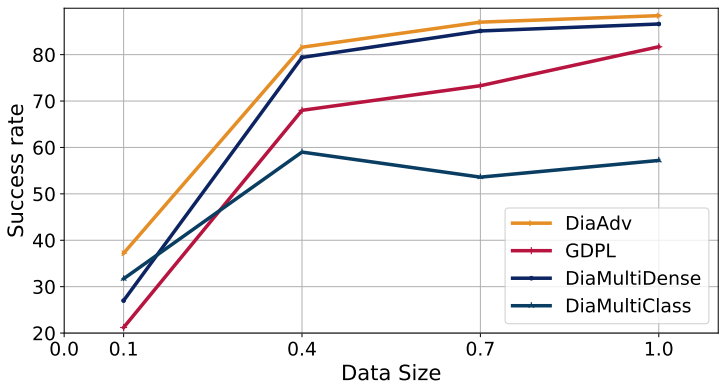


Figure 5.3: The performance gain between the pre-trained and their corresponding adversarial learning models with different amounts of expert dialogues.

by applying adversarial learning methods to the corresponding pre-trained models [5]. When the success rates of *DiaMultiClass* with MultiWoZ(0.4) and MultiWoZ(1.0) are both around 60%, deploying *GDPL* manages to bring 10% performance gain. The performance difference can be caused by the improved quality of the reward model. Conversely, if the reward model has no access to a sufficient amount of expert behaviors, it has little clue how the expert dialogues should look like. This can lead to poor reward signals for the policy network. We can see it in the case of *GDPL* that the success rate drops to 21% while the pre-trained model can achieve a 31% success rate on MultiWoZ(0.1). The performance gain between *DiaMultiDense* and *DiaAdv* is not so remarkable with respect to success rate compared to the gain between *DiaMultiClass* and *DiaAdv*. However, *DiaAdv* does help to reduce the dialogue turns while improving the success rate as shown in Table 5.3. We can regard *DiaAdv* as a promising method to fine-tune the *DiaMultiDense* to explore more potential dialogue states.

How sensitive are adversarial learning to pre-trained dialogue policy? We explore how pre-trained dialogue policies affect the final performance of adversarial learning-based dialogue agents. We first use supervised learning to pre-train the dialogue policies of *GDPL* and *DiaAdv* respectively with different training epochs. As

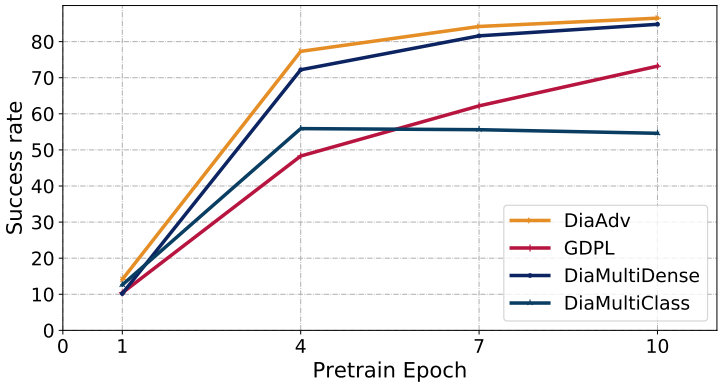


Figure 5.4: The performance gain between the pre-trained and their corresponding adversarial learning models with different amounts of pre-training epochs.

shown in Figure 5.4, the performance gain between the pre-trained dialogue policy and the corresponding adversarial are limited. With respect to *GDPL*, it even degenerates the original performance of the pre-trained policy when the starting points are relatively low. In other words, the main contributions to the adversarial dialogue agents come from the supervised learning stage; it is challenging for the dialogue agents to achieve the same performance without a promising pre-trained dialogue policy.

[5] *DiaAdv* is the adversarial extension of *DiaMultiDense* while *GDPL* is the adversarial extension of *DiaMultiClass*.

5.7 Conclusion

In this chapter, we proposed two supervised learning approaches and one adversarial learning method to train the dialogue policy for TDSs without building user simulators. The proposed methods can achieve state-of-the-art performance suggested by existing approaches based on RL and adversarial learning. However, we have demonstrated that our methods require fewer training efforts, namely the domain knowledge needed to design a user simulator and the intractable parameter tuning for RL or adversarial learning. Our findings have questioned if the full potential of supervised learning for dialogue PL has been exerted and if RL methods have been used in the appropriate TDS scenarios.

From a research perspective, since we are not dealing with real users, we can simply say that the answer to RQ4 is *"Yes"* in specific setups, meaning that we are making progress in applying reinforcement learning to dialogue policy learning under certain conditions. Different advanced RL methods for dialogue policy have been studied in the last few years and they help us to exploit the potential usages of RL-method in TDSs. In terms of the practical ability of these proposed RL-methods, the question remains open until we deploy the specific method to a dialogue system dealing with real users.

In both supervised training setup and adversarial training setup, high-quality human-human dialogue collections are vital components. The data-driven reward functions in Chapter 2,3 also strongly rely on expert demonstrations. However, in most cases, only the interaction logs between users and interactive systems are available and it is most likely that the collected interaction logs are noisy and suboptimal. So it is challenging but also promising to recover reward functions from noisy user behaviors. Besides, inferring the reward functions without actively collecting feedback in an adversarial training schema could also be an interesting direction. In the next chapter, more discussions about the current data-driven optimization framework will be given.

6 Conclusions

This chapter concludes this dissertation by revisiting our research questions from Chapter 1 (Section 1.1) and discussing our main findings (Section 6.1), and sketching directions for future research (Section 6.2). We focus on the main findings and general lessons, additional detailed findings are in the conclusion sections of the individual chapters.

6.1 Main Findings

The work included in this dissertation emerged from the following research challenge: *how to mine objectives directly from user interactions and use them to optimize the system.* In a series of empirical studies, we investigated several concrete research questions, which we will discuss next in turn.

6.1.1 Optimizing interactive systems with data-driven objectives

We started by addressing the following question:

RQ1 Can interactive systems be optimized using objectives recovered from user interactions directly?

The short answer is *"Yes"*. To answer this question in more detail, we first modeled user-system interactions using a Markov Decision Process (MDP). Different from previous work on modeling user interactions with interactive systems, we regarded the user as the agent while treating the interactive system as the stochastic environment. First, we suggest that users' behavior can be described by an unknown utility function which they tend to maximize through their interactions. Further, we assume a user's incentive to interact with the system is that he is getting rewards after each action. Treating an interactive system as a changeable and programmable environment is novel and practical because we have complete control of the interactive systems as system designers. Secondly, we formalized the objectives of optimizing interactive systems based on a restored user reward function which has taken into account the user preferences. After obtaining the objective, we proposed a novel algorithm, Interactive System Optimizer (ISO), that iterates between optimizing the interactive system for the current inferred objective; and let the user adapt to the new system behavior. This process

repeats until both the user and system policies converge. The objective formulation in the second step and interactive system optimization in the last step are happening simultaneously:

1. inferring an objective function directly from data, namely unlabeled trajectories of user interactions with the system;

2. iteratively optimizing the system according to the recovered data-driven objective as shown in Figure 2.1.

Therefore, in the whole process, no domain knowledge is required. To verify the feasibility of the proposed framework, we conducted two different simulated interactive setups, namely:

1. a tabular-based world with a finite set of states and actions; and

2. a neural network-based world with continuous state and action spaces.

In the second setup, the agent, the environment, and the reward function are all represented with separate neural networks because they have huge potential to represent the complex decision-making process for diverse interactive systems. In both experimental setups, we showed that the proposed framework can robustly improve user satisfaction while optimizing interactive systems.

The proposed interactive system optimizer can be viewed as a theoretical guideline. Hence, the solutions for real applications have required some tuning and adaptation. Regardless of the theoretical framework or practical application, the key idea, data-driven objectives, is a constant factor in this thesis. Next, we considered experiments with practical applications of interactive systems, namely dialogue systems.

6.1.2 Optimizing open-domain dialogue systems with data-driven objectives

We have started our investigation on applying ISO strategies to real-world applications by considering open-domain dialogue systems. Therefore, the second research question we studied in this dissertation is:

RQ2 Can data-driven reward functions be used to successfully improve open-domain dialogue systems?

To verify the effectiveness of the suggested data-driven reward functions, we started by investigating the drawbacks of existing optimization methods, namely adversarial training in dialogue generation. Due to the sparse and unstable reward signal from a poor discriminator, dialogue policy training suffers from mode collapse, leading to redundant and generic responses. We extended the adversarial dialogue generation approach to an adversarial imitation learning solution, which incorporates an entropy regularization term to the generation objective function. This addition could alleviate the problem of mode collapse. Then we adopted adversarial inverse reinforcement learning to train an open-domain dialogue generation model. The recovered reward function managed to provide a more precise reward for dialogue policy training. Especially for

the reward model, it acts at the word level, which takes as input the dialogue context, the newly generated word by the generator and returns the reward for the given dialogue context-action pair. With respect to the evaluation of generated responses, we proposed two human-evaluation settings along with several automatic evaluation metrics. Our experimental results demonstrated that our models can generate more high-quality responses and achieve higher overall performance than the state-of-the-art (SOTA) methods.

Our main finding for RQ2 is that by exploring the usage of data-driven reward functions, we proved that making use of data-driven reward function for response generation in open-domain dialogue systems is promising and helpful.

6.1.3 Optimizing task-oriented dialogue systems with data-driven objectives

By answering RQ2, we demonstrated the effectiveness of data-driven objectives in open-domain dialogue generation. Meanwhile, some researchers are investigating the potential of applying adversarial training to Task-oriented dialogue systems (TDSs), where multi-turn interactions are involved and long-term action influence between different dialogue turns should be considered. By going through all these methods, we found we were limited to policy gradient methods, such as REINFORCE and PPO, for the dialogue policy learning because alternating training between the dialogue agent and the reward model (or discriminator) is essential in an adversarial setup. Furthermore, the alternating training schema for the dialogue agent and the reward model can easily get stuck in local optima or result in mode collapse. There are some advanced off-policy Reinforcement Learning (RL) methods that have been proposed in the last few decades, such as Deep Q-learning (DQN), but these methods have so far been missing from data-driven system optimization. Therefore, our third research question was:

RQ3 Can off-policy RL methods benefit from data-driven objectives in dialogue policy learning for TDSs?

To apply data-driven objectives together with off-policy RL methods in dialogue policy learning, we decomposed the vanilla adversarial training into two sequential steps. We proposed to train a data-driven reward function and a dialogue policy *consecutively*, rather than in an alternating manner [83, 133]. Then we incorporate the trained dialogue reward model into the RL process of dialogue policy learning. In terms of the reward function training, we utilize an auxiliary dialogue state generator where the loss from the reward model can be directly backpropagated to the generator. It should be noted that this dialogue state generator is only used for reward training as a state explorer and it has no correlation with the dialogue agent in traditional adversarial dialogue training. Although we also have an adversarial training step in the whole optimization pipeline, it happens in the first step and will not hurt the dialogue policy in the second step. More importantly, the transfer from a SeqGAN [156] to a vanilla GAN [37] setting, leads us to a differentiable optimization environment. In a SeqGAN setup, the policy gradient method is essential to deliver the update signal from the discriminator to the dialogue agent. In contrast, in the vanilla GAN, the discriminator can directly backpropagate

the update signal to the generator due to the strait and close connection between the discriminator and generator.

Our main findings for RQ3 are two-fold. First, once we obtain the reward function, we are free to incorporate it in different reinforcement learning processes, including off-policy and on-policy methods, to guide dialogue policy learning. Since the learned reward function will not be updated during the dialogue policy training, we can also alleviate the problem of mode collapse in vanilla adversarial dialogue policy training setups. Secondly, finding is the potential of storing knowledge from existing domains into the reward function and then transferring it to new domains via RL. The knowledge is stored when the reward model is trained to distinguish human-generated dialogues from machine-generated ones, where the common information contained in high-quality human-generated dialogues are distilled. The stored information will be reused in the format of a reward signal during the training in an unseen domain. We verified the success of the proposed solution within a popular task-oriented dialogue system. We demonstrated that the proposed methods achieve remarkable performance, in terms of task success, as well as the potential to transfer knowledge from previously utilized task domains to new ones.

So far, we have considered two application scenarios of data-driven objectives:

- open-domain dialogue systems (RQ2);
- task-oriented dialogue systems (RQ3).

We have demonstrated the promising prospect of optimizing interactive systems with data-driven objectives. At the same time, we also need to recognize potential factors that could result in difficulties in applying this technique. All the frameworks and solutions introduced above rely on the use of deep RL methods. These methods require the model to actively interact with its environment and explicit feedback from the environment should be supplied. Considering TDSs as an example, the dialogue system predicts actions based on the dialogue policy, and the user feedback (the *reward signal*), which is provided when the dialogue is finished, is utilized to adjust the initial dialogue policy [21, 101, 149]. As it is not practical to ask for explicit user feedback for each dialogue during policy training, different strategies have been proposed to design a rule-based user simulator along with a reward function that can approximate the real *reward function*, which exists only in each user's mind. Ideally, we want a user simulator that can mimic real users and have realistic human-like behavior. However, this requires strong domain knowledge and becomes intractable when scaled to complex dialogue scenarios, such as dialogue systems dealing with multiple domains and diverse user behaviors. For example, if the task domain is complex and hard to solve, is it easier to design and maintain a complicated rule-based user simulator than to build a rule-based dialogue agent? Besides, adversarial training during reward learning may also require lots of effort. Solutions for interactive system optimization that involve reinforcement learning and inverse reinforcement learning have become relatively sophisticated. As another family of optimizing methods with data-driven objectives, what is the capability of supervised learning? These concerns, as relevant thinking of our studies into RQ2 and RQ3, push us to raise the following research question:

RQ4 Are we really making progress in applying only RL to dialogue policy learning

for TDSs?

This is not an easy question to answer due to the customized experimental setups (e.g., diverse domains, different action space, and user simulators) from existing work. Besides, the evaluation is not unified either, and this can bring unfairness in comparisons of the methods. Despite this, we think it is worth to conduct this investigation as far as possible. We first borrowed the idea of sequential classification from computer vision and proposed a dialogue action decoder within the framework of RNN to predict the appropriate actions sequentially. Furthermore, we revisited a traditional multi-label classification solution for dialogue policy learning. Based on it, we simply added dense layers and managed to improve the system significantly. The third method we presented is based on adversarial training between a dialogue agent and a discriminator and no user simulator is involved in the whole process. We have shown that all three methods can achieve comparable performance with the SOTA RL-based methods but with fewer efforts. Compared to existing RL based methods, we proposed a strategy that can eliminate designing a user simulator and sensitive parameter-tuning process while bringing a significant performance improvement with respect to a number of metrics. The absence of user simulators involved will largely reduce the required domain knowledge and supervised learning can lead to robust agent performance. However, we are not arguing that we should use supervised learning to replace reinforcement learning. There is a trade-off between designing a reliable user simulator and a high-quality labeled dialogue collection. For scenarios where collecting expert behaviors is feasible and affordable [11, 143, 148], RL may not be the best choice since traditional supervised learning solutions can achieve the same performance more easily. Indeed, off-line reinforcement learning is also a choice when a realistic user simulator is not accessible. But research into offline reinforcement learning is just getting started and we may not be able to find suitable algorithms for specific applications. In some cases, if the data cannot be collected for privacy or financial reasons, RL is the solution. Then, system performance after deployed to real scenarios highly depends on how realistic the user behavior generated by the user simulator are.

Back to dialogue policy learning in TDSs, the answer to RQ4 depends on how we evaluate dialogue systems and the answer can vary with different focuses and application scenarios. Simply relying on automatic metrics, of course, is not convincing. Even though some work has started adopting human evaluation as additional quality measurement, the answer is still not clear. For example, in many cases, human annotators are just responsible for judging the quality of the already generated dialogues between a dialogue agent and a user simulator, which means the human annotators are still not in the interaction loop. In another word, it is tricky to judge the performance of a dialogue agent in real-life scenarios when only the interaction histories of the same agent interacting with a user simulator are given. Therefore, a set of unified and reliable evaluation metrics for dialogue policy management is highly in demand. We presented some preliminary attempts of using the recovered reward functions to evaluate interactive systems in Chapter 3 and 4. This line of work is still at the very beginning and more concrete solutions should be investigated.

From a research perspective, since we are not dealing with real users, we can simply say that the answer to RQ4 is *"Yes"* in specific setups, meaning that we are making

progress in applying reinforcement learning to dialogue policy learning under certain conditions. Different advanced RL methods for dialogue policy have been studied in the last few years and they help us to exploit the potential usages of RL-method in Task-oriented dialogue system (TDS). In terms of the practical ability of these proposed RL-methods, the question remains open until we deploy the specific method to a dialogue system dealing with real users. In Chapter 5, we show that traditional supervised learning methods can achieve more stable and higher performance with fewer efforts, such as the domain knowledge required to design a user simulator and the intractable parameter tuning in reinforcement learning. Our findings demonstrate the value of rethinking the role of RL and supervised learning in optimizing TDSs. Applying RL methods to dialogue policy learning without considering the situations in real TDSs is not helpful to the progress of building more advanced dialogue systems.

6.2 Limitations and Future Directions

The work in Chapter 2 is our proposal for formulating interactive system optimization with data-driven objectives, while Chapter 3, 4, and 5 serve as the practical applications on the basis of ideas from Chapter 2. In this section, we will discuss two main limitations existing in the current optimization framework along with possible solution directions, namely:

- Section 6.2.1 discusses limitations that come along with building data-driven reward functions; and
- Section 6.2.2 provides overviews of possible limitations while working with user simulators in TDSs.

For a more detailed description of the limitations of each research question, readers can refer to the contents in the corresponding chapter.

6.2.1 Data-driven reward functions

As shown in Chapter 2, the final performance of the optimized system strongly relies on the quality of the recovered reward function. Recovering a reliable reward function requires a large number of high-quality user-system interaction trajectories. This could be the main difficulty preventing utilizing data-driven reward functions in applications dealing with real-world users. The dialogue trajectories used for reward learning in Chapter 4 and 5 are from a collection of human-human interactions that require a significant investment. Furthermore, a real-world interactive system usually serves different users. This makes collecting expert demonstrations with crowd-sourcing platforms more challenging because the collections should take into account different user preferences and behavior.

The most popular methods to recover a reward function are Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) and its more advanced extension, Adversarial Inverse Reinforcement Learning (AIRL). Since the scalability of vanilla MaxEnt-IRL is quite limited, AIRL is a more common choice for recovering rewards. However, this method is conducted within the framework of adversarial training and

this could potentially result in local optima and mode collapse, which are the inherent downside of adversarial training.

Possible solutions are exploring less data-sensitive reward recovery methods. Meanwhile, personalized user preferences should also be taken into account during reward learning. For example, user profile information can be incorporated in the state space or action space.

6.2.2 Interacting with user simulators

In the two-step optimization framework in Chapter 2, the user is involved in the whole pipeline, from the reward recovery step to the system optimization stage. During the reward recovery step, the user serves as the agent and has to actively execute actions to interact with the system. User feedback is essential for adversarial reward recovery. In terms of the next step, system optimization, the user serves as the environment to provide necessary responses for the state transitions with RL-based methods. Since it is not practical to have humans in the training step, most existing work replaces the real user with a rule-based user simulator. This is also the case for the TDSs in Chapter 4 and 5, where user simulators are essential for data-driven or RL approaches. The potential issue of this solution is that the user simulator may not be able to mimic real human behaviors. The behavior gap existing between user simulator and human behaviors will bring further issues to the evaluation and deployment of the optimized interactive system. These problems may have a negative impact on dialogue policy management in TDSs. Additionally, a unified and reliable evaluation method for dialogue agents is missing. Though human evaluation could be an alternative choice, this would be expensive and time-consuming. To have a more realistic user simulator, lots of domain knowledge is required and it is intractable when user behaviors are getting extremely complex (e.g., a system serving a huge user group and each individual has different preferences).

In recent years, off-line RL has attracted a lot of interest. The advantage of this type of approach is that it is not necessary to actively collect new interaction trajectories with the environment anymore. In our case, we can skip the user simulator in the second step of the optimization framework proposed in Chapter 2, and in the off-policy dialogue agent training in Chapter 4. Alternatively, we directly optimize the system with given historical interaction trajectories. However, we should recognize that the study of off-line deep RL just started several years ago and we may not be able to find suitable algorithms for the specific application yet. Furthermore, if we can extend off-line RL to off-line Inverse Reinforcement Learning (IRL), we will omit the user simulator utilized in the reward recovery step in Chapter 2 and 4 as well. The situation for off-line IRL is more challenging since this area is still virgin land that has not been explored.

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, pages 1–8. ACM, 2004. (Cited on pages 15 and 16.)

[2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. (Cited on pages 14 and 16.)

[3] F. Argelaguet, L. Hoyet, M. Trico, and A. Lécuyer. The role of interaction in virtual embodiment: Effects of the virtual hand representation. In *VR*, pages 3–10. IEEE, 2016. (Cited on pages 1 and 11.)

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. (Cited on pages 2 and 72.)

[5] L. Azzopardi. Modelling interaction with economic models of search. In *SIGIR*, pages 3–12. ACM, 2014. (Cited on pages 12 and 14.)

[6] R. E. Banchs. Movie-dic: a movie dialogue corpus for research and development. In *ACL*, pages 203–207, 2012. (Cited on page 43.)

[7] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, pages 1171–1179, 2015. (Cited on page 60.)

[8] L. E. Blume, S. Durlauf, and L. E. Blume. *The New Palgrave Dictionary of Economics*. Palgrave Macmillan Manchester, 2008. (Cited on page 12.)

[9] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov. A neural click model for web search. In *WWW*, pages 531–541, 2016. (Cited on pages 1 and 11.)

[10] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, pages 182–189, 2011. (Cited on pages 15 and 16.)

[11] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic. Multiwoz: A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026, 2018. (Cited on pages 58, 68, 73, and 85.)

[12] B. Burchfiel, C. Tomasi, and R. Parr. Distance minimization for reward learning from scored trajectories. In *AAAI*, pages 3330–3336. AAAI Press, 2016. (Cited on page 20.)

[13] H. Chen, X. Liu, D. Yin, and J. Tang. A survey on dialogue systems: recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35, 2017. (Cited on pages 53 and 67.)

[14] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *WSDM*, pages 456–464, 2019. (Cited on page 12.)

[15] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. (Cited on page 70.)

[16] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, pages 4299–4307, 2017. (Cited on page 15.)

[17] J. W. Crandall. Towards minimizing disappointment in repeated games. *Journal of Artificial Intelligence Research*, 49:111–142, 2014. (Cited on page 12.)

[18] C. Cui, W. Wang, X. Song, M. Huang, X.-S. Xu, and L. Nie. User attention-guided multimodal dialog systems. In *SIGIR*, pages 445–454. ACM, 2019. (Cited on pages 1 and 12.)

[19] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. In *SIGIR*, pages 65–74. ACM, 2017. (Cited on pages 1, 11, and 14.)

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. (Cited on page 65.)

[21] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016. (Cited on pages 1, 11, 14, 53, 54, 68, and 84.)

[22] A. Drutsa, G. Gusev, and P. Serdyukov. Engagement periodicity in search engine usage: Analysis and its application to search quality evaluation. In *WSDM*, pages 27–36, 2015. (Cited on page 12.)

[23] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016. (Cited on pages 14 and 16.)

[24] G. Dupret and M. Lalmas. Absence time and user engagement: evaluating ranking functions. In *WSDM*, pages 173–182, 2013. (Cited on page 12.)

[25] L. El Asri, R. Laroche, and O. Pietquin. Reward shaping for statistical optimisation of dialogue management. In *SLSP*, pages 93–101. Springer, 2013. (Cited on page 20.)

[26] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks,

inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016. (Cited on pages 15 and 16.)

[27] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, pages 49–58, 2016. (Cited on pages 2, 15, 16, 20, 37, and 39.)

[28] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, pages 357–368, 2017. (Cited on page 12.)

[29] J. L. Fleiss and J. Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement*, 33(3):613–619, 1973. (Cited on page 48.)

[30] G. Forgues, J. Pineau, J.-M. Larchevêque, and R. Tremblay. Bootstrapping dialog systems with word embeddings. In *NeurIPS, MML-NLP Workshop*, volume 2, 2014. (Cited on page 45.)

[31] S. Fox, K. Karnawat, M. Mydland, S. T. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168, 2005. (Cited on page 12.)

[32] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017. (Cited on pages 2, 3, 15, 16, 20, 37, and 40.)

[33] M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863*, 2015. (Cited on page 45.)

[34] M. Gašić and S. Young. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40, 2014. (Cited on pages 3, 53, and 67.)

[35] M. Gašić, N. Mrkšić, P.-h. Su, D. Vandyke, T.-H. Wen, and S. Young. Policy committee for adaptation in multi-domain spoken dialogue systems. In *ASRU*, pages 806–812. IEEE, 2015. (Cited on page 73.)

[36] M. Gombolay, R. Jensen, J. Stigile, T. Golen, N. Shah, S.-H. Son, and J. Shah. Human-machine collaborative optimization via apprenticeship scheduling. *Journal of Artificial Intelligence Research*, 63:1–49, 2018. (Cited on page 12.)

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. (Cited on pages 3, 55, and 83.)

[38] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954. (Cited on pages 56 and 71.)

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. (Cited on pages 14 and 16.)

[40] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. (Cited on pages 14 and 16.)

[41] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, pages 4565–4573, 2016. (Cited on pages 15, 16, 40, 60, 69, and 74.)

[42] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, pages 4565–4573, 2016. (Cited on pages 3 and 48.)

[43] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR*, pages 251–263. Springer, 2011. (Cited on page 14.)

[44] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*, pages 183–192. ACM, 2013. (Cited on pages 12 and 14.)

[45] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval Journal*, 16(1):63–90, 2013. (Cited on page 14.)

[46] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. (Cited on pages 56, 59, 71, and 74.)

[47] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. (Cited on page 12.)

[48] H. J. Jeon, S. Milli, and A. D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *arXiv preprint arXiv:2002.04833*, 2020. (Cited on pages 12, 14, and 15.)

[49] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161, 2005. (Cited on page 12.)

[50] D. Kelly. Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval*, 3(1–2):1–224, 2009. (Cited on page 12.)

[51] D. Kelly. When effort exceeds expectations: A theory of search task difficulty. In *ECIR, SCST Workshop*, 2015. (Cited on page 12.)
[52] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on page 44.)
[53] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. (Cited on page 57.)
[54] J. Kiseleva and M. de Rijke. Evaluating personal assistants on mobile devices. *arXiv preprint arXiv:1706.04524*, 2017. (Cited on pages 1 and 11.)
[55] J. Kiseleva, E. Crestan, R. Brigo, and R. Dittel. Modelling and detecting changes in user satisfaction. In *CIKM*, pages 1449–1458, 2014. (Cited on page 15.)
[56] J. Kiseleva, J. Kamps, V. Nikulin, and N. Makarov. Behavioral dynamics from the SERP's perspective: what are failed SERPs and how to fix them? In *CIKM*, pages 1561–1570, 2015. (Cited on page 15.)
[57] J. Kiseleva, K. Williams, A. H. Awadallah, I. Zitouni, A. Crook, and T. Anastasakos. Predicting user satisfaction with intelligent assistants. In *SIGIR*, pages 45–54. ACM, 2016. (Cited on pages 1, 11, and 12.)
[58] J. Kiseleva, K. Williams, J. Jiang, A. H. Awadallah, I. Zitouni, A. Crook, and T. Anastasakos. Understanding user satisfaction with intelligent assistants. In *CHIIR*, pages 121–130, 2016. (Cited on pages 1 and 11.)
[59] M. Kosinski, D. Stillwell, and T. Graepe. Private traits and attributes are predictable from digital records of human behavior. *PNAS*, 110:5802–5805, 2013. (Cited on page 12.)
[60] M. Kutlu, V. Khetan, and M. Lease. Correlation and prediction of evaluation metrics in information retrieval. *arXiv preprint arXiv:1802.00323*, 2018. (Cited on page 14.)
[61] S. Lee, Q. Zhu, R. Takanobu, X. Li, Y. Zhang, Z. Zhang, J. Li, B. Peng, X. Li, M. Huang, et al. Convlab: Multi-domain end-to-end dialog system platform. *arXiv preprint arXiv:1904.08637*, 2019. (Cited on pages 55, 59, 69, and 77.)
[62] J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, and S. Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018. (Cited on pages 12, 14, and 15.)
[63] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000. (Cited on pages 12 and 14.)
[64] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. (Cited on pages 12, 14, and 16.)
[65] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, pages 173–184, 2016. (Cited on pages 12, 14, and 16.)
[66] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119, 2016. (Cited on pages 2 and 37.)
[67] J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016. (Cited on page 45.)
[68] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202, 2016. (Cited on pages 1, 2, 11, 12, 14, and 37.)
[69] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky. Adversarial learning for neural dialogue generation. In *EMNLP*, pages 2157–2169, 2017. (Cited on pages 2, 5, 37, 38, 40, 41, 44, and 50.)
[70] X. Li, Z. C. Lipton, B. Dhingra, L. Li, J. Gao, and Y.-N. Chen. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016. (Cited on pages 3, 4, 6, 53, and 69.)
[71] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017. (Cited on pages 2, 14, 54, and 68.)
[72] Z. Li and M. de Rijke. The impact of linkage methods in hierarchical clustering for active learning to rank. In *SIGIR*, pages 941–944, 2017.
[73] Z. Li, J. Kiseleva, M. de Rijke, and A. Grotov. Towards learning reward functions from user interactions. In *ICTIR*, pages 289–292, 2017. (Cited on page 12.)
[74] Z. Li, J. Kiseleva, A. Agarwal, and M. de Rijke. Learning data-driven objectives to optimize interactive systems. In *NeurIPS LIRE 2019 Workshop: Learning with Rich Experience: Integration of Learning Paradigms*, 2019.
[75] Z. Li, J. Kiseleva, and M. de Rijke. Dialogue generation: From imitation learning to inverse reinforcement learning. In *AAAI*, volume 33, pages 6722–6729, 2019. (Cited on pages 1, 11, 15, 37, and 54.)

[76] Z. Li, J. Kiseleva, A. Agarwal, M. de Rijke, and R. W. White. Optimizing interactive systems via data-driven objectives. *Journal of Artificial Intelligence Research*, 2020. Submitted. (Cited on page 11.)

[77] Z. Li, J. Kiseleva, and M. de Rijke. Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. In *EMNLP findings*, 2020. (Cited on page 67.)

[78] Z. Li, S. Lee, B. Peng, J. Kiseleva, M. de Rijke, J. Li, S. Shayandeh, and J. Gao. Guided dialogue policy learning without adversarial learning in the loop. In *EMNLP findings*, 2020. (Cited on pages 15, 53, and 68.)

[79] Z. Li, J. Kiseleva, and M. de Rijke. Generating coherent and informative responses with backward-reasoning in open-domain dialogue systems. In *NAACL*, 2021. Submitted.

[80] Z. Li, D. Park, J. Kiseleva, and S. Lee. Estimating user satisfaction level for multi-turn dialogues. In *NAACL*, 2021. Submitted.

[81] X. Lin, S. C. Adams, and P. A. Beling. Multi-agent inverse reinforcement learning for certain general-sum stochastic games. *Journal of Artificial Intelligence Research*, 66:473–502, 2019. (Cited on page 15.)

[82] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *AAAI*, 2018. (Cited on pages 12, 14, 54, and 68.)

[83] B. Liu and I. Lane. Adversarial learning of task-oriented neural dialog models. In *SIGDIAL*, pages 350–359, 2018. (Cited on pages 3, 4, 5, 6, 54, 55, 69, 74, and 83.)

[84] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132, 2016. (Cited on page 45.)

[85] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, 2016. (Cited on page 12.)

[86] F. Lovett. Rational choice theory and explanation. *Rationality and Society*, 18(2):237–272, 2006. (Cited on page 12.)

[87] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL*, pages 1116–1126, 2017. (Cited on page 15.)

[88] J. Luo, X. Dong, and H. Yang. Learning to reinforce search effectiveness. In *ICTIR*, pages 271–280. ACM, 2015. (Cited on page 14.)

[89] J. Luo, X. Dong, and H. Yang. Session search by direct policy learning. In *ICTIR*, pages 261–270. ACM, 2015. (Cited on pages 1, 12, and 14.)

[90] J. Mitchell and B. Shneiderman. Dynamic versus static menus: an exploratory comparison. *ACM SigCHI Bulletin*, 20(4):33–37, 1989. (Cited on page 13.)

[91] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. (Cited on pages 6 and 51.)

[92] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015. (Cited on pages 14, 16, and 60.)

[93] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2012. (Cited on page 14.)

[94] M. Monfort, A. Liu, and B. Ziebart. Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In *AAAI*, pages 3672–3678, 2015. (Cited on page 15.)

[95] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670. ACM, 2000. (Cited on pages 15, 16, and 18.)

[96] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *SIGCHI*, pages 597–606, 2007. (Cited on page 13.)

[97] D. Odijk, E. Meij, I. Sijaranamual, and M. de Rijke. Dynamic query modeling for related content finding. In *SIGIR*, pages 33–42. ACM, 2015. (Cited on page 14.)

[98] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002. (Cited on pages 12 and 45.)

[99] B. Peng, X. Li, L. Li, J. Gao, A. Celikyilmaz, S. Lee, and K.-F. Wong. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*,

2017. (Cited on pages 12, 14, 54, and 68.)
[100] B. Peng, X. Li, J. Gao, J. Liu, Y.-N. Chen, and K.-F. Wong. Adversarial advantage actor-critic model for task-completion dialogue policy learning. In *ICASSP*, pages 6149–6153. IEEE, 2018. (Cited on pages 1, 11, 54, and 69.)
[101] B. Peng, X. Li, J. Gao, J. Liu, and K.-F. Wong. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *ACL*, volume 1, pages 2182–2192, 2018. (Cited on pages 6, 53, 68, and 84.)
[102] J. Perner and B. Lang. Development of theory of mind and executive control. *Trends in cognitive sciences*, 3(9):337–344, 1999. (Cited on pages 1 and 12.)
[103] O. Pietquin. Inverse reinforcement learning for interactive systems. In *Workshop on Machine Learning for Interactive Systems*, pages 71–75. ACM, 2013. (Cited on page 15.)
[104] A. H. Qureshi, B. Boots, and M. C. Yip. Adversarial imitation via variational inverse reinforcement learning. In *ICLR*, 2019. (Cited on pages 15 and 16.)
[105] A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, et al. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*, 2018. (Cited on page 1.)
[106] N. D. Ratliff, D. Silver, and J. A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009. (Cited on pages 15 and 16.)
[107] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike. Learning human objectives by evaluating hypothetical behavior. *arXiv preprint arXiv:1912.05652*, 2019. (Cited on pages 12 and 14.)
[108] A. Ritter, C. Cherry, and B. Dolan. Unsupervised modeling of twitter conversations. In *NAACL*, pages 172–180, 2010. (Cited on pages 1, 37, and 44.)
[109] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *EMNLP*, pages 583–593, 2011. (Cited on page 45.)
[110] A. I. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, and A. Oh. Creating natural dialogs in the carnegie mellon communicator system. In *ECSCT*, 1999. (Cited on page 2.)
[111] V. Rus and M. Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162, 2012. (Cited on page 45.)
[112] S. Russell. Learning agents for uncertain environments. In *COLT*, pages 101–103. ACM, 1998. (Cited on pages 15 and 16.)
[113] T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for information science*, 26(6):321–343, 1975. (Cited on page 12.)
[114] T. Saracevic, P. Kantor, A. Y. Chamis, and D. Trivison. A study of information seeking and retrieving. *Journal of the American Society for Information science*, 39(3):161–176; 177–196; 197–216, 1988. (Cited on page 12.)
[115] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, pages 149–152, 2007. (Cited on pages 3, 53, and 59.)
[116] T. Schnabel, P. N. Bennett, and T. Joachims. Shaping feedback data in recommender systems with interventions based on information foraging theory. In *WSDM*, pages 546–554, 2019. (Cited on pages 1 and 11.)
[117] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, chess and Shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019. (Cited on pages 14 and 16.)
[118] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. (Cited on pages 3, 14, 16, 28, 54, 60, and 73.)
[119] A. Sepliarskaia, J. Kiseleva, F. Radlinski, and M. de Rijke. Preference elicitation as an optimization problem. In *RecSys*, pages 172–180, 2018. (Cited on pages 1 and 11.)
[120] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784, 2016. (Cited on pages 2, 37, 43, 44, and 50.)
[121] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301, 2017. (Cited on pages 2, 37, 45, 47, and 50.)
[122] S. K. Seyed Ghasemipour, S. S. Gu, and R. Zemel. Smile: Scalable meta inverse reinforcement

learning through context-conditional policies. In *NeurIPS*, pages 7881–7891. 2019. (Cited on pages 15 and 16.)

[123] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *ACL*, volume 1, pages 1577–1586, 2015. (Cited on pages 1, 2, 37, and 50.)

[124] G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005. (Cited on page 14.)

[125] Z. Shi, X. Chen, X. Qiu, and X. Huang. Towards diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018. (Cited on page 38.)

[126] H.-Y. Shum, X.-d. He, and D. Li. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26, 2018. (Cited on page 1.)

[127] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. (Cited on pages 12, 14, and 16.)

[128] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, pages 196–205, 2015. (Cited on pages 2, 37, and 50.)

[129] P.-H. Su, M. Gasic, N. Mrkšić, L. M. R. Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young. On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*, volume 1, pages 2431–2441, 2016. (Cited on pages 53 and 68.)

[130] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*, 2017. (Cited on pages 3, 53, and 67.)

[131] S.-Y. Su, X. Li, J. Gao, J. Liu, and Y.-N. Chen. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *EMNLP*, 2018. (Cited on pages 12, 14, 54, and 68.)

[132] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. (Cited on pages 12, 14, 15, 16, 17, and 18.)

[133] R. Takanobu, H. Zhu, and M. Huang. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In *EMNLP*, pages 100–110, 2019. (Cited on pages 3, 4, 5, 6, 14, 15, 54, 55, 60, 64, 69, 74, and 83.)

[134] J. Teevan. How people recall, recognize, and reuse search results. *ACM Transactions on Information Systems (TOIS)*, 26(4):1–27, 2008. (Cited on page 13.)

[135] M. ter Hoeve, R. Sim, E. Nouri, A. Fourney, M. de Rijke, and R. W. White. Conversations with documents: An exploration of document-centered assistance. In *CHIIR*, pages 43–52, 2020. (Cited on pages 1 and 11.)

[136] H. R. Varian. Economics and search. In *ACM SIGIR Forum*, volume 33, pages 1–5. New York, 1999. (Cited on page 12.)

[137] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. (Cited on pages 14 and 16.)

[138] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*, 1997. (Cited on pages 1, 3, 4, and 68.)

[139] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016. (Cited on pages 14 and 16.)

[140] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016. (Cited on page 73.)

[141] H. Wei, F. Zhang, N. J. Yuan, C. Cao, H. Fu, X. Xie, Y. Rui, and W.-Y. Ma. Beyond the words: Predicting user personality from heterogeneous information. In *WSDM*, pages 305–314. ACM, 2017. (Cited on page 12.)

[142] J. 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. (Cited on pages 1 and 67.)

[143] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015. (Cited on pages 68 and 85.)

[144] R. W. White. *Interactions with Search Systems*. Cambridge University Press, 2016. (Cited on pages 1,

11, and 14.)
[145] R. W. White, I. Ruthven, and J. M. Jose. Finding relevant documents using top ranking sentences: an evaluation of two alternative schemes. In *SIGIR*, pages 57–64, 2002. (Cited on page 13.)
[146] R. W. White, I. Ruthven, J. M. Jose, and C. V. Rijsbergen. Evaluating implicit feedback models using searcher simulations. *ACM Transactions on Information Systems (TOIS)*, 23(3):325–361, 2005. (Cited on page 24.)
[147] J. D. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007. (Cited on pages 53 and 67.)
[148] J. D. Williams, M. Henderson, A. Raux, B. Thomson, A. Black, and D. Ramachandran. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124, 2014. (Cited on pages 68 and 85.)
[149] J. D. Williams, K. Asadi, and G. Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*, 2017. (Cited on pages 1, 11, 14, 53, 54, 68, and 84.)
[150] K. Williams, J. Kiseleva, A. Crook, I. Zitouni, A. H. Awadallah, and M. Khabsa. Is this your final answer? evaluating the effect of answers on good abandonment in mobile search. In *SIGIR*, 2016. (Cited on pages 1 and 11.)
[151] K. Williams, J. Kiseleva, A. C. Crook, I. Zitouni, A. H. Awadallah, and M. Khabsa. Detecting good abandonment in mobile search. In *WWW*, pages 495–505, 2016. (Cited on pages 1, 11, and 12.)
[152] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. (Cited on pages 3 and 54.)
[153] M. Wulfmeier, P. Ondruska, and I. Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015. (Cited on page 2.)
[154] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W.-Y. Ma. Topic aware neural response generation. In *AAAI*, volume 17, pages 3351–3357, 2017. (Cited on pages 1 and 37.)
[155] E. Yilmaz, M. Verma, N. Craswell, F. Radlinski, and P. Bailey. Relevance and effort: An analysis of document utility. In *CIKM*, pages 91–100, 2014. (Cited on page 12.)
[156] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017. (Cited on pages 3, 55, and 83.)
[157] Z. Yu, L. Nicolich-Henkin, A. W. Black, and A. Rudnicky. A wizard-of-oz study on a non-task-oriented dialog systems that reacts to user engagement. In *Proceedings of the 17th annual meeting of the Special Interest Group on Discourse and Dialogue*, pages 55–63, 2016. (Cited on page 1.)
[158] J. Y. Zhang and A. D. Dragan. Learning from extrapolated corrections. In *ICRA*, pages 7034–7040, 2019. (Cited on page 15.)
[159] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *SIGKDD*, pages 1040–1048, 2018. (Cited on pages 12 and 14.)
[160] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li. Drn: A deep reinforcement learning framework for news recommendation. In *WWW*, pages 167–176, 2018. (Cited on page 12.)
[161] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364, 2017. (Cited on pages 14 and 16.)
[162] B. Ziebart, A. Dey, and J. A. Bagnell. Probabilistic pointing target prediction via inverse optimal control. In *IUI*, pages 1–10. ACM, 2012. (Cited on page 15.)
[163] B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010. (Cited on page 24.)
[164] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438. AAAI Press, 2008. (Cited on pages 15, 16, 20, 39, and 40.)
[165] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010. (Cited on page 39.)
[166] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. Juplter: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 8(1):85–96, 2000. (Cited on page 2.)
[167] V. W. Zue and J. R. Glass. Conversational interfaces: Advances and challenges. *Proceedings of the IEEE*, 88(8):1166–1180, 2000. (Cited on page 2.)

Summary

Interactive systems are becoming more and more popular to assist humans to obtain information or complete tasks effectively and efficiently. Building such systems requires a lot of effort, and understanding what users want and designing corresponding optimization objectives are some of the key components. The reliability of hand-crafted objectives strongly relies on the amount of domain knowledge incorporated in them. It becomes intractable to obtain such objectives given more and more complex systems and user behavior. A new solution to optimizing interactive systems is in high demand and this leads us to the overall research topic of this dissertation: optimizing interactive systems with data-driven objectives.

In the first part of this thesis, we explore how to optimize interactive systems without handcrafting objectives in a more general setup, where the system and the user are both simulated. We model the interactions between users and systems with two separate MDPs: the first one is built to infer user objectives by recovering user reward functions from demonstrations, and the second one is for optimizing the system with the inferred objectives in a reinforcement learning setup. Importantly, our solution requires no domain knowledge and is thus even applicable when prior knowledge is absent. The results in two simulated experimental setups show that the proposed two-step optimization framework can robustly improve the interactive system's performance.

In the second part of the thesis, we first utilize the idea of data-driven objectives for two types of interactive systems: open-domain dialogue systems and task-oriented dialogue systems. In terms of open-domain dialogue systems, we start by investigating the drawbacks of existing data-driven optimization methods, namely adversarial training in dialogue generation. We make use of the causal entropy regularization term to stabilize the adversarial training process. Furthermore, we adopt adversarial inverse reinforcement learning to train a dialogue generation model. Especially for the reward model, it acts at the word level and returns the reward for the given dialogue context-action pair. With respect to dialogue policy learning in task-oriented dialogue systems, off-policy based reinforcement learning methods are absent in the solutions of data-driven objectives. To make off-policy methods also benefit from data-driven objectives in dialogue policy learning, we propose to decompose vanilla adversarial dialogue policy training into two consecutive steps: (1) training a data-driven reward function with an auxiliary dialogue state generator, and (2) incorporating the inferred reward function to different reinforcement learning processes, including off-policy and on-policy methods, to guide dialogue policy learning. Furthermore, the trained data-driven reward function can serve as a bridge to transfer knowledge from existing domains to new domains via reinforcement learning.

In addition to exploring the promising usage scenarios of data-driven objectives, we also investigate the limitations and potential problems of current deep reinforcement learning based solutions for dialogue policy learning in task-oriented dialogue systems. Solutions for dialogue system optimization that involve reinforcement learning and inverse reinforcement learning have become relatively sophisticated. Are we really making progress in applying reinforcement learning to dialogue policy learning? We demonstrate how (1) traditional supervised learning together with (2) a simulator-free adversarial learning method can be used to achieve performance comparable to state-

of-the-art reinforcement learning methods. Our findings have questioned if the full potential of supervised learning for dialogue policy learning has been exerted and if reinforcement learning methods have been used in the appropriate task-oriented dialogue system scenarios.

Samenvatting

Interactieve systemen genieten toenemende populariteit bij het helpen van mensen om effectief en efficiënt informatie te vinden en taken te volbrengen. Het bouwen van dergelijke systemen vereist veel inspanning, waarbij het begrijpen van wat gebruikers willen, en de bijbehorende optimalisatie doelen opstellen, een centrale rol spelen. De betrouwbaarheid van manueel opgestelde optimalisatie-doelen (objectives) is sterk afhankelijk van de domeinkennis die in de doelen worden vastgelegd. Het wordt snel onhaalbaar om dergelijke optimalisatie doelen vast te stellen voor complexe systemen en complexe gedragspatronen. Een nieuwe oplossing voor het optimaliseren van interactieve systemen is daarom sterk gewenst en vormt het onderwerp van dit proefschrift: het optimaliseren van interactieve systemen met data-gedreven optimalisatie-doelen.

In het eerste deel van dit proefschrift verkennen wij in een generieke opstelling hoe interactieve systemen kunnen worden geoptimaliseerd zonder handmatige optimalisatiedoelen, waarbij zowel de gebruiker als het systeem worden gesimuleerd. We modelleren de interacties tussen gebruikers en systemen in twee aparte MDPs: De eerste achterhaalt de beloningsfunctie voor het systeem uit demonstraties. De tweede optimaliseert het systeem met de gegeven optimalisatie-doelen in een Reinforcement Learning context. De belangrijkste observatie is dat deze oplossing geen a priori domeinkennis vereist en daardoor kan worden toegepast in omstandigheden waar domeinkennis onbeschikbaar is. Resultaten in twee simulatie experimenten laten zien dat deze twee-staps methode een robuuste verbetering aan interactieve systemen kan bieden.

In het tweede deel van dit proefschrift gebruiken wij de data-gedreven optimalisatiedoel methode voor twee typen interactieve systemen: open-domein dialoog systemen en taakgeoriënteerde dialoog systemen. Voor open-domein dialoog systemen starten we met een onderzoek naar de problemen bij bestaande data-gedreven optimalisatie methoden, namelijk *adversarial learning* bij dialoog-generatie. We gebruiken een causale entropie regularisatie term om het oppositionele systeem te stabiliseren. Daarnaast gebruiken we inverse *adversarial reinforcement learning* om het dialoog generatie model te trainen. Dit werkt vooral voor de beloningsfunctie, die op woordniveau, op basis van de dialoog context, een woord kiest en daarbij een beloning genereert voor de gegeven dialoog context-actie paren. Bij de dialoog-beleid leertaak in het taakgeoriënteerde dialoog systeem zijn de off-policy acties géén onderdeel van de data-gedreven optimalisatie-doelen. Om off-policy acties te kunnen gebruiken bij data-gedreven optimalisatie doelen van dialoog-systemen, stellen wij voor om normaal *adversarial learning* op te breken in twee opeenvolgende stappen: (1) een data-gedreven beloningsfunctie trainen met een aanvullende dialoog generator en (2) de geleerde beloningsfunctie integreren in andere oppositionele leerprocessen met zowel on-policy als off-policy methodes om het leerproces te begeleiden. Daarnaast kunnen de getrainde beloningsfuncties worden gebruikt als brug om kennis over te brengen van bekende domeinen naar nieuwe domeinen via versterkingsleren.

Ter aanvulling van de verkenning van veelbelovende toepassingsscenario's van data-gedreven optimalisatiedoelen kijken we ook naar de beperkingen en problemen van huidige *deep-learning reinforcement learning* gebaseerde oplossingen voor taakgeörienteerde dialoogsystemen. Oplossingen voor dialoogsysteem-optimalisatie door middel van *reinforcement learning* en inverse *reinforcement learning* zijn relatief hoog

ontwikkeld. Boeken we echt vooruitgang in de toepassing van versterkingsleren in dialoogsystemen? We laten zin hoe (1) traditionele begeleid leren (*supervised learning*) samen met (2) simulatorvrij oppositioneel trainen kan worden gebruikt om vergelijkbare resultaten te halen als state-of-the-art *reinforcement learning* methodes. Onze bevindingen werpen de vraag op of het volledige potentieel van begeleid leren voor dialoogsystemen is benut en of *reinforcement learning* methodes zijn gebruikt in geschikte scenario's voor taakgeoriënteerde dialoog systemen.