Reinforcement Learning and Reward Estimation for Dialogue Policy Optimisation

Pei-Hao Su

Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

Queens' College April 2018

To my family

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements. This dissertation contains fewer than 45,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 35 figures. Some of the material included in this thesis has been published in the International Speech Communication Association (Su et al., 2015a), the Special Interest Group on Discourse and Dialogue (Su et al., 2017, 2015b), the IEEE Automatic Speech Recognition and Understanding Workshop (Vandyke et al., 2015), the Association for Computational Linguistics (Su et al., 2016b), the arXiv preprint (Su et al., 2016a), the NIPS Deep Reinforcement Learning Symposium (Casanueva et al., 2017), and Computer Speech and Language (Su et al., 2018).

Pei-Hao Su

April 2018

Acknowledgements

First and foremost, I would like to thank my supervisor, Steve Young, for the constant encouragements and mentorship throughout my PhD study in the past three years. Steve, I enjoy every conversation with you, your wisdom and passion for research have constantly reshaped my drafted ideas into accomplished research projects. I am very grateful to you not only for guiding me about advances in dialogue systems, but also for teaching me how to think critically. To my advisor, Milica Gašić, thank you for all the valuable discussions and your insights in dialogue management and Gaussian processes, which led to a very successful work. I appreciate my viva examiners, Dr Rich Turner and Professor Phil Blunsom, for the helpful comments on my thesis. I would also like to thank Lin-Shan Lee, my undergrad and master advisor, for introducing me to this exciting research field.

Special thanks go to Tsung-Hsien and Nikola. I have grown so much from our healthy competition and mindful discussion. Tsung-Hsien, my ten-year friend since undergrad, your enthusiasm on research has always motivated me. Nikola, my best Balkan friend, I enjoy the great adventure and nice food in Tel Aviv and Belgrade. My PhD study could not have been so enjoyable without you two.

I am privileged to work with lots of inspiring minds at the Dialogue Systems Group. I would like to thank my close collaborators, David and Paweł. David, my first-year study can not be more prolific without your dedication and feedbacks. Paweł, thank you for staying up several midnights with me and always stays positive and reliable. I thank Dongho, Pirros, Matt, Iñigo, Stefan and Lina for making my lab life so pleasant. In addition, I also thank Rachel and Katherine for their excellent administrative support as well as Anna and Patrick for their computing support.

Outside the work, I thank friends at the Taiwanese Society and Queens' college who have enriched my Cambridge life. Profound gratitude goes to all the members at 63 Rustat: Chia-Chi, Jinco, Fiona, Henry and Pierre. I have had a lot of cheerful memories with you all, especially during my thesis writing. I also thank Lin, my best neighbour and the most talented person, for your endless delicious food and life advices. I thank my friends: Xie, Yishu, Matko, Kris, Marek, Daniela and many others. Furthermore, I sincerely thank Cambridge Trust and the Ministry of Education in Taiwan for funding me throughout my studies.

To Yi-Chun, thank you for your patience, your love, and your support these years. Because you always understood what is at the heart of me.

Finally, I would like to thank my family for all their love and encouragement. For my parents, Chien-Cheng and Yin-Hsueh, who have raised me with endless love and supported me in all my pursuits. For my sister, Yu-Ting, who always shares wonderful and joyful life experience with me.

Thank you all for your support throughout my unforgettable PhD journey.

Abstract

Modelling dialogue management as a reinforcement learning task enables a system to learn to act optimally by maximising a reward function. This reward function is designed to induce the system behaviour required for goal-oriented applications, which usually means fulfilling the user's goal as efficiently as possible. However, in real-world spoken dialogue systems, the reward is hard to measure, because the goal of the conversation is often known only to the user. Certainly, the system can ask the user if the goal has been satisfied, but this can be intrusive. Furthermore, in practice, the reliability of the user's response has been found to be highly variable. In addition, due to the sparsity of the reward signal and the large search space, reinforcement learning-based dialogue policy optimisation is often slow. This thesis presents several approaches to address these problems.

To better evaluate a dialogue for policy optimisation, two methods are proposed. First, a recurrent neural network-based predictor pre-trained from off-line data is proposed to estimate task success during subsequent on-line dialogue policy learning to avoid noisy user ratings and problems related to not knowing the user's goal. Second, an on-line learning framework is described where a dialogue policy is jointly trained alongside a reward function modelled as a Gaussian process with active learning. This mitigates the noisiness of user ratings and minimises user intrusion. It is shown that both off-line and on-line methods achieve practical policy learning in real-world applications, while the latter provides a more general joint learning system directly from users.

To enhance the policy learning speed, the use of *reward shaping* is explored and shown to be effective and complementary to the core policy learning algorithm.

Furthermore, as deep reinforcement learning methods have the potential to scale to very large tasks, this thesis also investigates the application to dialogue systems. Two sample-efficient algorithms, trust region actor-critic with experience replay (TRACER) and episodic natural actor critic with experience replay (eNACER), are introduced. In addition, a corpus of demonstration data is utilised to pre-train the models prior to on-line reinforcement learning to handle the cold start problem. Combining these two methods, a practical approach is demonstrated to effectively learn deep reinforcement learning-based dialogue policies in a task-oriented information seeking domain.

Overall, this thesis provides solutions which allow truly on-line and continuous policy learning in spoken dialogue systems.

Table of contents

List of figures

List of tables

CHAPTER 1

Introduction

A spoken dialogue system (SDS) allows human-computer interaction using natural language and speech. However, since human-computer interaction differs from natural human conversation, teaching a machine to understand and converse is non-trivial and challenging. This study encloses a broad spectrum of research fields such as computational linguistics, engineering design, behaviour learning, and artificial intelligence. This chapter starts with an overview of an SDS and its existing limitations and then presents the contributions and a roadmap of this dissertation.

1.1 Overview

Research into SDSs is regarded by many as one of the most challenging and long-term tasks. In his *imitation game* (Turing, 1950), Turing attempted to define the standard of 'intelligence' for a machine by asking if a human participant could differentiate it from a human *through conversation*. He also suggested that an intelligent machine should simulate the learning process of a human child. This idea has a close connection to the modern machine learning techniques used to build an SDS.

The development of SDSs has been greatly improved since the concept was established. An early computer programme called ELIZA (Weizenbaum, 1966) was created in the 60's, which imitated the language of a psychotherapist with simple word reordering and convinced some people to mistake her for a human. With the recent theoretical advances and improvement in practical deployment, the possibility of talking naturally to machines becomes much more plausible. Depending on the purpose, an SDS can be broadly divided into two categories: *chat-oriented systems* which aim to converse with users and provide reasonable contextually relevant responses, and *task-oriented systems* which assist users to achieve specific goals (e.g. find hotels, movies or bus schedules). A chat-oriented SDS is expected to talk about everything in an *open domain* setting. It is not necessary to have a well-defined goal or intention. Conversations on social media platforms such as Facebook, Twitter and Reddit are typically categorised in such scenarios. However, the potentially infinite number of topics and language variation are the bottlenecks to creating reasonable responses. The challenge of such systems centres on maintaining a reasonable conversation similar to talking to a rational person. On the other hand, the task-oriented SDS typically operates in a *closed domain* setting where the possible inputs and outputs are restricted since the system aims to achieve a very specific goal. It is typically designed according to a structured ontology (or a database schema) which defines the domain that the system can talk about. When talking to these systems, the users often have a clear goal in mind and do not expect the system to handle all cases. This type of SDS can be found in many practical applications, such as call centres and help desks which answer basic questions from customers (Georgila et al., 1998; Lowe et al., 2015), website navigation tools (Chai et al., 2001), pedagogical agents which assist learners to improve their problem solving and language skills (Su et al., 2013, 2015c), interactive games which engage players in situated scenarios (Zielke et al., 2009), and modern intelligent personal assistant software such as Apple's Siri [1], Amazon Echo [2], Google Assistant [3], Microsoft Cortana [4], Facebook M [5] and Slack [6].

Deploying such automated agents can significantly reduce human labour costs. Hence, SDS has great potential and is fascinating for both academia and industry. However, despite its many benefits, existing SDSs are far from perfect. Most commercial systems still utilise human-designed dialogue flow charts to handle all of the use cases, which are hard to maintain and hard to scale to larger domains. To mitigate this, data-driven techniques and machine learning algorithms have been exploited and proved to be promising for improving robustness and extendibility of spoken dialogue technology (Young et al., 2013b). The goal is to build a fully statistical SDS with robust spoken language understanding and generation capability and automate the design and planning of each system reply in a dialogue. This thesis focuses on task-oriented systems and uses an example application of tourist information seeking, where users of the system can search for and ask about venues like restaurants in a town according to their constraints (e.g. by price-range or cuisine type). The long-term goal of this research direction is to develop an end-to-end statistical framework that robustly learns from and interacts with human users.

1.2 Motivation

An SDS is typically built upon various components: a speech recogniser, a spoken language understanding module, a dialogue manager, a language generator, and a speech synthesiser. This thesis is concerned broadly with the design of a dialogue manager which determines the system response at each turn. More specifically,

[1] http://www.apple.com/ios/siri/
[2] http://www.amazon.com/oc/echo
[3] https://assistant.google.com/
[4] https://www.microsoft.com/en-us/windows/cortana
[5] https://www.wired.com/2015/08/facebook-launches-m-new-kind-virtual-assistant/
[6] https://slack.com/apps/category/At0MQP5BEF-bots

Fig. 1.1 An example of a task-oriented dialogue with a pre-defined task and the evaluation results.

the main focus is on determining a suitable learning signal from humans, a *reward*, and making the best use of it for learning a real-world dialogue manager on-line with human users. Generally, a 'reward' from the human can be determined in numerous ways, including by voice/language, facial expression, or numerical ratings. Here the focus is on human ratings.

The development of a statistical dialogue manager involves data collection and model training. In a task-oriented setting, the collection of human-machine conversation often conforms to a pre-defined task as shown in the left part of Figure 1.1. This objective defines a clear goal for the system to reach and clear evaluation metric for measurement. In this scenario, the recruited subjects are given a specific task to follow. The collected data are further used to train the model to either 'mimic' the human responses via supervised learning (Bordes et al., 2017; Wen et al., 2017a) or plan its own dialogue flow using reinforcement learning (RL) (Sutton and Barto, 1999). However, since the users may have (slightly) different goals in their minds and their satisfaction rating may be unreliable, the *objective* evaluation results according to the task may not match the *subjective* rating from the user. Figure 1.1 illustrates an example of the mismatch between the two measurements. Here the user did not ask for the phone information and thought the dialogue was successful as the system answered all concerns. On the other hand, this task was not completed according to the description, because the phone number slot was not marked. This mismatch hinders system performance, as the conversational data is inconsistent with the labels. In addition, this label is often sparse, because it is set as an overall evaluation only when the dialogue ends; this requires the learning system to explore more until it finds the (sub-)optimal solutions. To summarise, the two main challenges of reward setting in statistical dialogue management are as follows:

1. precisely determining the 'goodness' of the human-machine interaction.
2. efficiently exploiting the (sparse) reward signal to increase the speed of the learning process.

In addition, the current state-of-the-art methods such as the Gaussian process (GP)-based RL policy are mainly non-parametric and do not scale well. In contrast, deep RL methods offer the potential to scale to very large tasks but have several problems: slow learning and poor performance in the early stages of training (cold start). To summarise, this thesis has two main emphases: designing efficient and robust reward estimation and improving the learning speed of deep RL methods. This thesis explores these issues and presents solutions to address them.

1.3 Contributions and Outline

The dissertation is organised as follows:

- **Chapter 2 - Overview of Spoken Dialogue Systems**
  The general concepts and architecture of an SDS are described in this chapter, with detailed descriptions of each component.
- **Chapter 3 - Dialogue Policy Optimisation and Reward Estimation**
  This chapter reviews background knowledge and summarises related works

on dialogue policy optimisation and reward estimation, with a specific focus on RL approaches. The benefits and current challenges of these works are enumerated, which leads to the work presented in the following chapters to address the problems.

- **Chapter 4 - Policy Learning with Off-line Reward Estimator**
This chapter investigates the use of a RNN model to infer the dialogue success information from an off-line corpus. The resulting reward model enables practical policy learning with real users without checking the task information in a tourist information seeking domain. The proposed model is also extended to scale beyond one domain by making the model and corresponding input features domain-independent. Part of the research work has been presented in the following publications (Su et al., 2015a; Vandyke et al., 2015):

  - **P.-H. Su**, D. Vandyke, M. Gašić, D. Kim, N. Mrkšić, T.-H. Wen, and S. Young, "Learning from Real Users: Rating Dialogue Success with Neural Networks for Reinforcement Learning in Spoken Dialogue Systems". *In Proc of INTERSPEECH 2015.*
  - D. Vandyke, **P.-H. Su**, M. Gašić, N. Mrkšić, T.-H. Wen, and S. Young, "Multi-Domain Dialogue Success Classifiers for Policy Training". *In Proc of ASRU 2015.*

- **Chapter 5 - Policy Learning with On-line Active Human Reward Estimator**
This chapter proposes an on-line learning framework whereby the dialogue policy is jointly trained alongside the reward model via active learning with a GP model. This GP operates on a continuous space dialogue representation generated in an unsupervised fashion using a RNN encoder-decoder. The experimental results demonstrate that the proposed framework can significantly reduce data annotation costs and mitigate noisy user feedback in dialogue policy learning. Some of these contributions have been presented in the following publication (Su et al., 2018, 2016b):

- **P.-H. Su**, M. Gašić, N. Mrkšić, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "On-line active reward learning for policy optimisation in spoken dialogue systems". *In Proc of ACL 2016.* (**Best Student Paper**)
- **P.-H. Su**, M. Gašić, and S. Young, "Reward Estimation for Dialogue Policy Optimisation". *In Computer Speech and Language 2018.*

- **Chapter 6 - Accelerating Policy Learning with Reward Shaping**
The sparse nature of reward in RL slows down learning since many interactions are needed before getting any reward. This chapter explores the use of *reward shaping* to enrich the intermediate reward during the dialogue to speed up learning. The experiment shows that the learning speed increased whilst maintaining similar performance. This research work has been presented in the publication (Su et al., 2015b):

  - **P.-H. Su**, D. Vandyke, M. Gašić, N. Mrkšić, T.-H. Wen, and S. Young, "Reward Shaping with Recurrent Neural Networks for Speeding up On-Line Policy Learning in Spoken Dialogue Systems". *In Proc of SIGDIAL 2015.*

- **Chapter 7 - Learning Sample-efficient Deep RL Policies**
Deep reinforcement learning methods have significant potential for dialogue policy optimisation. However, they suffer from slow learning speed and poor performance in the early learning stages. This chapter first explores the use of experience replay with off-policy learning to enrich the update information (with a batch of data) to the models and combines it with two better gradient update methods: trust region optimisation (TRACER) and natural gradient descent (eNACER). Second, to mitigate the cold start issue, a corpus of demonstration data is utilised to pre-train the models prior to on-line RL. Combining these two approaches, a practical approach to learn

deep RL- based dialogue policies is presented. This research work has been published as international papers (Casanueva et al., 2017; Su et al., 2016a, 2017):

- **P.-H. Su**, M. Gašić, N. Mrkšić, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "Continuously learning neural dialogue management". *In arXiv preprint 2016.*
- **P.-H. Su**, P. Budzianowski, S. Ultes, M. Gašić, and S. Young, "Sample-efficient Actor-Critic Reinforcement Learning with Supervised Data for Dialogue Management". *In Proc of SIGDIAL 2017.*
- I. Casanueva*, P. Budzianowski*, **P.-H. Su***, N. Mrkšić, T.-H. Wen, S. Ultes, L. Rojas-Barahona, S. Young, and M. Gašić,"A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management". *In NIPS Deep Reinforcement Learning Symposium 2017.*

- **Chapter 8 - Conclusions**
  Concluding remarks of the main contributions are summarised in this chapter with several potential directions and extensions.

In summary, the thesis contribution is divided into two main sections. Chapters 4 and 5 focus on building a robust reward model to learn the dialogue policy from human users. Chapters 6 and 7 focus on enhancing the learning efficiency of the dialogue policy.

CHAPTER 2

Overview of Spoken Dialogue Systems

A *spoken dialogue* is a conversational exchange between two or more people where the primary medium is speech. To simplify the task, two assumptions are made. First, a dialogue involves only two participants. Second, the dialogue consists of a sequence of *turns*, where each turn includes an utterance from both participants with fixed order.

The architecture of an SDS has not been standardised. Nevertheless, most systems follow the pipeline design demonstrated by Pieraccini and Huerta (2005), which is illustrated in 2.1. The details of each component are described in the following sections.

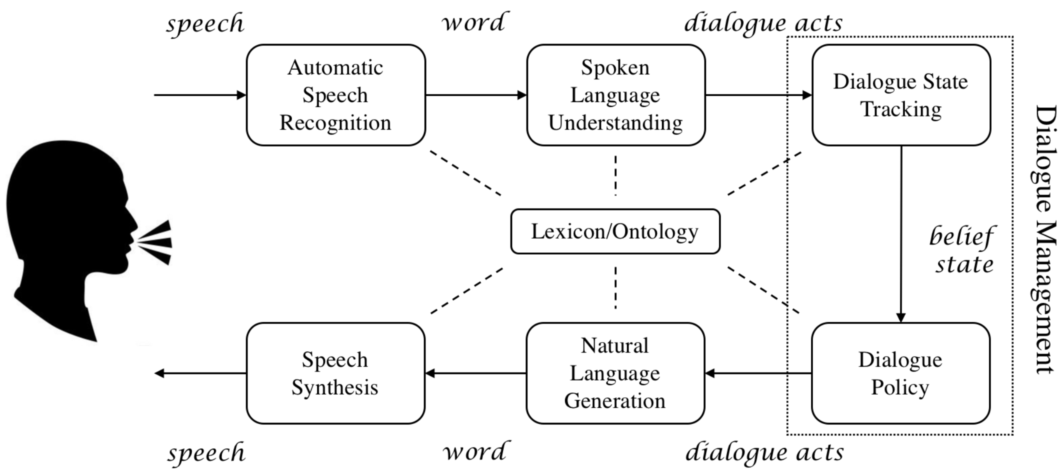


Fig. 2.1 The pipeline architecture of a spoken dialogue system.

2.1 Automatic Speech Recognition

The automatic speech recognition (ASR) component transcribes the audio signal into written form output. This output is generally represented as an *N-best list* of the most probable sentences, or as a *word lattice* (or *confusion network* in a more restricted form Mangu et al. (2000)) formed in a directed graph where each edge has a word and its associated weight allows the path probability to be calculated. In this work, an ASR system generating an N-best list is adopted. This is discussed more in the following sections.

Most ASR systems consist of an acoustic model, a language model (LM), and a lexicon. These are inputted with a raw audio signal or a pre-processed feature vector, such as Mel frequency cepstral coefficients (MFCCs). The acoustic model computes the most probable sequence of phonemes, the basic units of speech, which best represent the input vectors. This phoneme sequence is then mapped to texts using a lexicon. One popular model is the hidden Markov model (HMM), where the state probabilities are modelled with a Gaussian mixture model (GMM). A comprehensive overview is presented in Gales and Young (2008). A more recent line of research has concentrated on the use of neural networks to replace the GMM and the approaches now represent the state-of-the-art (Dahl et al., 2012; Deng et al., 2013; Hinton et al., 2012; Waibel et al., 1989). On the other hand, the LM provides linguistic constraints to help the selection of correct words, and such models are traditionally implemented using n-gram models (Brown et al., 1992) or more recently with neural networks (Bengio et al., 2003; Chen et al., 2016; Jozefowicz et al., 2016; Mikolov et al., 2010).

To simplify the deployment process, current research is focussed on *End-to-end* ASR, aiming to combine the acoustic model and LM using RNNs (Bahdanau et al., 2016; Chan et al., 2016; Graves et al., 2013b; Hannun et al., 2014).

2.2 Spoken Language Understanding

Having converted the speech signal into words, the spoken language understanding (SLU) module aims at determining the underlying semantics of the utterance.

SLU broadly covers the research of domain detection, intent determination (Tur and Deng, 2011; Xu and Sarikaya, 2013), and slot filling (Chen et al., 2013; Mesnil et al., 2015a). In dialogue research, the semantics are typically represented in the form of a *dialogue act*[1]. For instance, the utterance:

'I plan to stay at a hotel in the city centre tonight and price is not my concern.'

can be formed as:

`inform(area=centre,price=dontcare,type=hotel).`

This form captures the intent of the sentence, `action=inform`, meaning that the user provides certain information, the actual slot-value pairs which specify the arguments of the act: `area=centre,price=dontcare`, and the domain: `type=hotel`. Note that this is an example of a flat-structural representation. Depending on the design of the ontology and application, there are more ways to represent a sentence such as relation-based semantic representation (Harispe et al., 2014) , which has richer structured concepts. This type of representation is closely related to the logic form in computational linguistics and general artificial intelligence. In this thesis, only limited domains are considered, hence flat dialogue acts are used. This form is also used as the conceptual representation of the output from the dialogue policy component in 2.4.

The simplest and most straightforward method to extract the meaning from a sentence is to design a set of rules that map from words or phrases to the concepts, called *rule-based semantic grammars* (Ward, 1994; Zue et al., 2000). However, these rules need to be expanded or re-designed when the domain coverage is changed; thus, it is not scalable. For more complex sentences with richer linguistic variations and ambiguities, more advanced techniques such as the use of context-free gram-

[1]Dialogue acts are a special form of speech act (Searle, 1969), representing the meaning of an utterance.

mars (CFG) (Charniak, 1997; Kwiatkowski et al., 2011; Zettlemoyer and Collins, 2012), methods that rely on inductive logic programming (Zelle and Mooney, 1993), or dependency parsing trees (Chen and Manning, 2014; Jurafsky and Martin, 2008; Nivre et al., 2007) have been investigated.

Statistical approaches to language understanding require a set of training data and their corresponding labels (Raymond and Riccardi, 2007). Most common labels are word level (aligned) or entire sentence level (unaligned). The Air Travel Information System (Price, 1990) is one of the commonly used datasets containing both aligned and unaligned labels for evaluating model performance. Widely known statistical models include *generative* approaches such as dynamic Bayesian network (DBN) (He and Young, 2006), discriminative models such as support vector machine (SVM) (Kate and Mooney, 2006), conditional random fields (CRF) (Jeong and Lee, 2008; Lafferty et al., 2001), and more recently neural network approaches (Hermann et al., 2015; Kalchbrenner et al., 2014; Mesnil et al., 2015a; Yao et al., 2014). When embedded in a pipelined SDS, the impact of SLU on the downstream performance has also been investigated (Li et al., 2017).

2.3 Dialogue State Tracking

A dialogue essentially comprises a sequence of interactions. It is, therefore, important to consider the previous context information and the current observation (either ASR or SLU output) from the user at any point of the conversation to determine what the user is talking about. The Markovian representation which summarises the current state of the dialogue is called the *dialogue state*. In contrast to standard SLU tasks which operate on single-turn decoding, the task of dialogue state tracking (DST) centres on the multi-turn determination of the user's intention. Probabilistic models often maintain a *belief state*, which is the distribution over all dialogue states (Rapaport, 1986).

Dialogue State Tracking (Henderson, 2015; Williams et al., 2016) has been intensively investigated by various research groups in recent years. It can be best shown by the five completed series of dialogue state tracking challenges to date (Henderson et al., 2014a,b; Kim et al., 2017, 2016; Williams et al., 2013). The challenges cover a wide spectrum of DST tasks, from single and multiple domain human-machine interactions, goal-changing scenarios, and human-human conversations.

Approaches to address this problem are generally similar and closely related to the methods for SLU tasks, including simple rule-based and statistical methods. The former utilises hand-crafted rules to update the dialogue state given the output from the SLU component (Wang and Lemon, 2013), and the latter learns these rules from the given data. Early statistical models employed generative models such as DBN to jointly model the current observations from SLU outputs and the dialogue state (Young et al., 2007), Bayesian update of dialogue state (BUDS) (Thomson et al., 2008) was one specific example. Later, discriminative models, which aim to calculate the conditional probability of the belief state given the dialogue history, have been found to generally outperform generative methods. Log-linear models such as maximum entropy linear model (Lee and Eskenazi, 2013; Williams, 2014) and CRF (Ren et al., 2013) have been shown to be effective in this task. Recently, neural networks have also been applied and achieved state-of-the-art results (Henderson et al., 2014c; Kadlec et al., 2014; Mrkšić et al., 2015, 2017; Perez and Liu, 2016).

In this work, the BUDS dialogue state tracker (Thomson and Young, 2010) is mainly adopted.

2.4 Dialogue Policy

Once the user's intention has been determined, the system must choose an appropriate reply. Serving as the first component for the system response as shown in the bottom right of Figure 2.1, the dialogue policy component controls the interaction between the system and the user and is central to the overall quality of the user experience. Its behaviour is determined by a dialogue policy, which maps the belief states (or dialogue state) to system actions.

To manage the dialogue interaction, the simplest approach is to hand-craft the dialogue flow. In this case, expert knowledge is usually utilised and the overall dialogue flow is formed in a tree-structured (or a finite state) format (Larsson and Traum, 2000; McTear, 1998; Sutton et al., 1996). However, this approach requires a large amount of human effort and needs further modification when new information which changes the domain's ontology and dynamic is added. In addition, the uncertainty created by noisy ASR and SLU often leads to a poor DST result, which brings the system to an incorrect state and causes it to react incorrectly and potentially generate an unreasonable system response.

It has, therefore, been suggested in recent years that statistical methods (Daubigney et al., 2012a; Gašić et al., 2014; Keizer et al., 2010b; Lemon and Pietquin, 2007) can resolve the shortcomings introduced by handcrafted approaches. Probabilistic models allow the uncertainty in the system's belief state to be mapped into a distribution over sets of system action. This improves the robustness and error recovery ability of the system under various noisy conditions. A dialogue policy could be cast as a classification task, where a set of corpus data is provided for the system to train on. In this approach, the system learns to select an action given the statistics of the particular state-action pairs in the corpus. Nevertheless, the corpus dataset often lacks sufficient coverage of all possible situations and the perfect imitation on the given corpus does not necessarily lead to a successful dialogue interaction. Thus, it is more adequate to view the dialogue interaction as a long-term planning task and optimise its action selection policy to achieve a higher success rate. To automatically obtain such policies, RL is often utilised.

When modelling the dialogue policy with RL approaches, the goal of the model is to select a sequence of system replies (actions) given the observed belief states to maximise the total reward; in this work, the reward is mainly determined by the dialogue success. If only the top hypothesis from the DST component is presented as the dialogue state input to the dialogue policy, the problem is modelled as a Markov decision process (MDP) (Levin et al., 1998; Singh et al., 2000). If multiple hypotheses are presented or a distribution over all dialogue states, a belief state, is given to the dialogue policy, the problem becomes a partially observable Markov decision process (POMDP) (Williams and Young, 2007a; Young, 2002). The details of applying an MDP and a POMDP to a dialogue policy is further explained in Section 3.1.

2.5 Natural Language Generation

Given the dialogue act selected from the dialogue policy, the natural language generation (NLG) component then maps it back to a sentence. For example, the dialogue act: `confirm(area=city centre)` can be mapped to: "You are looking for the place in the city centre, right?"

Rule-based (or template-based) NLG systems have been widely utilised because of its simplicity, robustness and high accuracy in limited domains. This method is adopted in this work, where a fixed set of hand-crafted rules is created to generate sentences. The main issues with these systems are their scalability to large domains and the lack of language variability on their output sentences (Stent et al., 2005).

Corpus-based systems, on the other hand, aim to learn from a set of corpus data to reduce the efforts on creating rules. Many such systems were inspired by the language models used in ASR. Such systems generate the desired sentences by considering the target intention and semantics. This includes the early work on a class-based n-gram LM approach proposed by Oh and Rudnicky (2000), the phrase-based NLG system based on factored LMs (Mairesse and Young, 2014), and recently some neural network approaches (Hu et al., 2017; Wen et al., 2015). On the other hand, Rieser and Lemon (2010) have proposed using RL to solve this problem.

2.6 Speech synthesis

The speech synthesis (SS) component converts the chosen text or the symbolic linguistic representation into speech. Many systems rely on the *concatenative* approach (Hunt and Black, 1996). Such systems hold a large database of short pre-recorded speech fragments from a single speaker and recombine a portion of these fragments to form an utterance. The shortcoming of this method is that the generated speech often sounds discontinuous, since the combined fragments might be recorded in different scenarios. Furthermore, it is difficult to modify the emotion of the voice or switch to a different speaker without recording a whole new set of short speech fragments.

Another widely used approach is the *parametric* SS (Zen et al., 2009), where the model is mainly parametrised by an HMM. This HMM-based SS is also used in this work. In this method, the information for generating the speech is stored in the model parameters, and the characteristics and contents of the speech can be easily controlled by the inputs. Such existing models typically pass their outputs through a *vocoder* (Holmes, 2001) to generate audio signals. In dialogue systems, this parametric method is also shown to learn dialogue context sensitive behaviours to provide more natural speech (Tsiakoulis et al., 2014a,b).

Furthermore, capitalising on the success of neural network models, the use of deep and recurrent neural networks have also been successfully applied to parametric SS (Ling et al., 2015; Zen et al., 2016, 2013). Recently, researchers have even applied neural networks to directly model the raw signal and yield more natural-sounding speech (Oord et al., 2016).

2.7 User Simulator

Ideally, when utilising an RL method to model the dialogue policy, the system would be optimised in interactions with real users. However, for real-world tasks, training a reasonable policy normally requires hundreds of thousands of dialogues, and the initial poor performance may result in a negative user experience. This makes learning from scratch with real users difficult. Furthermore, it is financially costly and time-consuming to collect large amounts of dialogue data. Hence, building a user simulator that can interact with the dialogue policy is very useful (Schatzmann et al., 2006). During the algorithm development process, the user simulator can be used to converse and train the system as many times as possible.

A user simulator is typically built upon a corpus of example dialogues, and it is expected to have three characteristics: (1) behave similar to a rational real user, (2) generate a coherent sequence of sentences/actions, and (3) generalise to new contexts. How to evaluate a user simulator remains an open question, and there is not yet a widely accepted metric to measure the correlation between the behaviour of the real user and the simulator (Pietquin and Hastie, 2013; Schatzmann et al., 2005). The user simulator can interact with the system in either the dialogue act level, the word level, or the speech level (Jung et al., 2009). In addition, to simulate noisy conditions, an error model is typically utilised to distort the output of the user simulator (Schatzmann et al., 2007b). A statistical user simulator is built upon a corpus of the example human dialogues.

To effectively encode the dialogue history and user goal, the agenda-based user simulator with parameters estimated from data as described in Keizer et al. (2010a); Schatzmann et al. (2007a) is used in this thesis. In this method, the user goal is decomposed into a set of slot-value pairs representing the requests and constraints and stored in a stack. An example of the user simulator is shown in Figure 2.2, where the left is the user goal and the right representing the user agenda of the turn-level user intentions in semantic level (dialogue act and slot-value pairs). An example dialogue in semantic level is also shown in 6.5.

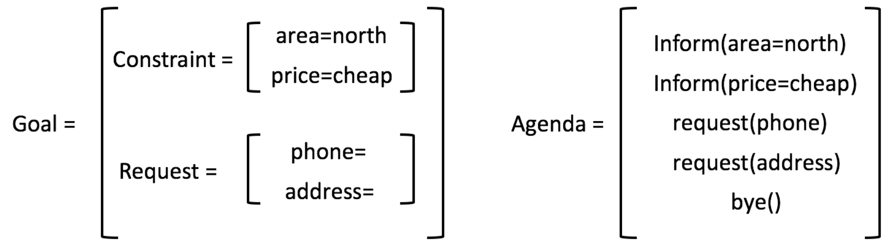


Fig. 2.2 An example of the user simulator.

Depending on the situation, the stack pushes or pops a user's intention during the interaction. This ensures consistency in the goal-oriented user behaviour across the entire conversation.

In addition to the agenda model in this work, there are many other ways to build a user simulator such as n-gram methods (Eckert et al., 1997), graph- based (Scheffler and Young, 2002), or a Bayesian approach (Pietquin and Dutoit, 2006). A sequence-to-sequence approach for training a user simulator has been recently proposed by Asri et al. (2016). This method views the user-input utterance to system-output response as a source-to-target sequence generation task, as inspired by the original sequence-to-sequence model (Sutskever et al., 2014). This model requires minimal feature engineering and can potentially generalise well with a large amount of labelled data.

CHAPTER 3

Dialogue Policy Optimisation and Reward Estimation

Dialogue can be viewed as a decision-making task. At each turn, the SDS first analyses and understands the user's utterance by the ASR and SLU components. Then it decides on what to say to the user given the entire dialogue history encoded by the DST. The dialogue policy is the core module that is responsible for controlling the flow of the dialogue. In Section 2.4, the introduction of rule-based and statistical approaches to dialogue policy were briefly described. Since statistical methods, especially RL, are generally more scalable for real-world SDS deployment, these are the focus of this chapter.

When developing a dialogue policy, two fundamental issues often appear:

1. How to implement a dialogue policy?

2. What is a suitable design criterion or objective?

In the following, literature reviews on these two questions are presented. Section 3.1 introduces the basics of RL and illustrates how it has been adopted for dialogue policy. Training a model via RL requires a clear definition of the learning objective, Section 3.2 therefore discusses various work on dialogue reward estimation and dialogue evaluation.

3.1 Reinforcement Learning for Dialogue Policy Optimisation

RL (Sutton and Barto, 1999) is a subfield of machine learning, concerned with how an agent (the machine) learns to interact with the *environment*. In any situation, the agent perceives an *observation* from the *environment* and determines which *action* to take and receives a *reward*. The goal of this agent is to find a sequence of actions which maximises the total (or expected) reward. A dialogue policy can be appositely modelled using an RL framework since the system can learn how to reply to the user given the transcribed dialogue context via the human-computer interactions. Many research results have shown that it is beneficial for dialogue applications (Janarthanam et al., 2011; Lee and Eskenazi, 2012; Young et al., 2013a), especially under the framework of POMDP. The formal definition of POMDPs and their application to dialogue are discussed in Section 3.1.1. The definition is followed by an overview of the training methods in Section 3.1.2 and the introduction of two main training approaches, value-based (Section 3.1.3) and policy-based (Section 3.1.4) methods. Finally, a taxonomy of RL training methods is illustrated in Section 3.1.5.

3.1.1 Dialogue as a Partially Observable Markov Decision Process

Partially observable MDPs are a generalisation of MDPs. It models an agent operating in a world where the system dynamics are decided by an MDP but without observing the underlying states. An MDP is a mathematical framework describing an agent interacting with a stochastic environment. It is formally represented by a tuple $\{S, A, T, R, \gamma\}$, which includes the set of all states S, the set of possible actions A, the Markovian state transition function T: $P(s_{t+1}|s_t, a_t)$, the reward function R defining the immediate reward $r(s_t,a_t)$, and γ as a geometric discount factor.

In the MDP setting, the state in the environment is fully observable. The goal is to find a policy π which selects an action at each state, $\pi : S \rightarrow A$. In principle, this policy aim to determine an action while viewing the entire history so far, including all previous observations, states, and actions. However, this is often intractable due to the high complexity of the long sequence. To handle this issue, the state is often assumed to satisfy the *Markov property* and depends only on its previous state.

The policy can be considered as stochastic, which is a conditional distribution $\pi(a|s)$, or deterministic, which is $\pi(s) = a$. It is optimised to maximise the expected discounted reward, R_t^π from time t, which is the sum of the discounted rewards over a potentially infinite horizon:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{3.1}$$

The discount factor $\gamma \in [0,1]$ is used to reduce the importance of rewards collected in later steps.

The MDP framework, however, is not well-suited for real-world SDS. Its assumption on the fully observability of states prevents it from coping with the corrupted ASR and SLU outputs due to different levels of noisy conditions and the inherent ambiguity of the natural language. This leads to POMDPs, which offer a principled mathematical model for agents to act in a non-deterministic domain under partial observability, which are suitable for handling real-world sequential decision tasks.

A POMDP is determined by a tuple $\{\Omega, S, A, T, R, O, \gamma\}$, where $\{S, A, T, R, \gamma\}$ is the underlying MDP, Ω is the set of observations, and O defines observation probabilities: $P(o_{t+1}|s_t,a_t)$. Figure 3.1 shows the influence diagram of a POMDP. The connections between these notations and the dialogue setting are as follows.

- **Observation**: The agent inspects the noisy observation $o \in \Omega$ from the world. As described in section 2.2, the output of the SLU component is typically an N-

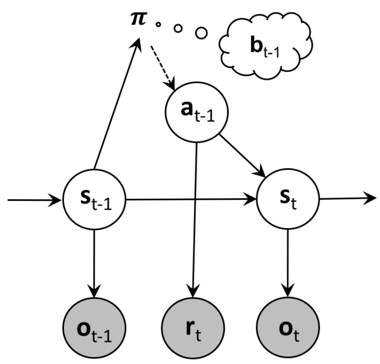


Fig. 3.1 POMDP influence diagram. Shaded nodes denote observable variables and non-shaded circles represent un-observed ones. Solid lines show direct influence, and the dashed line represent a distribution on the pointed variable.

best list of the scored semantic hypotheses and is what the agent understands from the user.

- **State**: A state $s \in S$ captures all related information in the environment (user). It is latent and inferred from the observation o. Since it often depends on the number of entities in the domain's ontology which is usually large-scale, the consequent dimensionality of the state space is a crucial concern in the real-world SDS. Efficient approximation techniques are therefore adopted to deal with its "curse of dimensionality".

- **Action**: The execution of the action $a \in A$ results in a state transition which then updates the agent's understanding of the environment. In the statistical paradigm of SDS, the collection of all possible replies from the system forms an action set. Since language has a high variability, a sentence such as '*You want a hotel that is in the north*' is often denoted by a higher-level representation from the SLU component such as `confirm(type=hotel,area=north)`. In some cases, this representation is further heuristically mapped to a summary action space such as `Confirm` for tractability.

- **Transitions**: In real world applications, the environment is dynamic, causing uncertainty in the effect of each action and resulting in stochastic transitions between states. A transition function $P(s_{t+1}|s_t,a_t)$ specifies the probability of arriving at state s_{t+1} when the agent performs action a_t from state s_t.

- **Observation Probability**: Probability $P(o_{t+1}|s_t,a_t)$ where the agent observes o_{t+1} after executing action a_t and reaching state s_t. This often models the accuracy of the systems' sensing system.

- **Reward**: A reward $r(s,a)$ is either stochastic or deterministic. It serves as an objective that directs the agent to learn a desirable behaviour. In an SDS, it is a crucial concern since measuring the quality of a dialogue is non-trivial and often involves human feedback. This issue is one of the main research topics in this thesis and is investigated in-depth in the following sections.

- **Discount Factor**: $\gamma \in [0,1]$ determines the effect of future outcomes on the current state s. In the SDS setting, it is typically set close to 1 since the dialogue length is finite and each dialogue turn has equal importance.

A schematic diagram of modelling dialogue management in the RL framework is illustrated in Figure 3.2. At each time step t, the agent gets an observation o_t from the user (environment). It is determined by the ASR, SLU and DST components. The current state s_t is then inferred from this observation o_t and the previous state s_{t-1}. Since states are unobservable, the system thus maintains a distribution over all possible states, called the *belief state* $b(s_t)$. Given the state space S, the belief space B is $[0,1]^{|S|}$. The initial distribution over all states is therefore $\mathbf{b}_0 = [b(s_0 = s^1),...,b(s_0 = s^{|S|})]^\top$. Based on the current belief $b(s_t)$, the agent takes an action a which is then transformed into natural speech using the NLG and TTS. When each action is taken, the agent receives a reward $r(s_t,a_t)$. This also causes the environment to transition to state s_{t+1} with probability $P(s_t,a,s_{t+1}) = P(s' = s_{t+1}|s = s_t, a = a_t)$.

The model in the POMDP dialogue framework generally learns to update the belief over dialogue state and determines a good policy (Thomson and Young, 2010). In recent years, these two parts have been decomposed into a DST and a

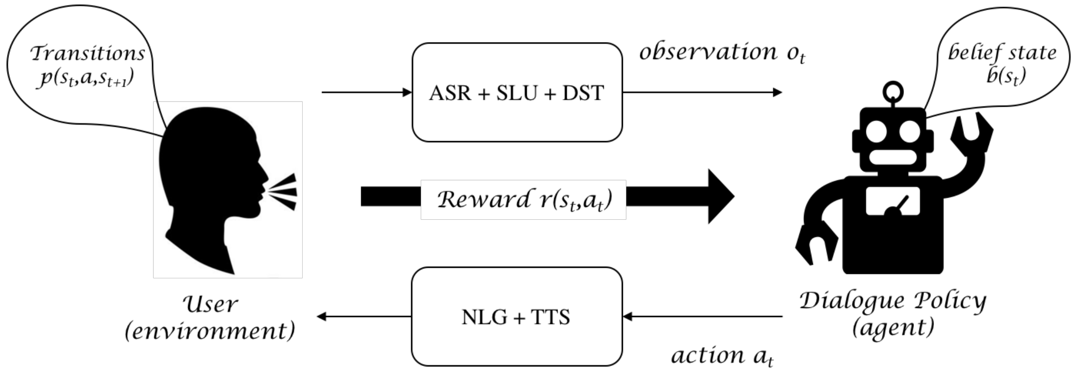


Fig. 3.2 Dialogue policy optimisation in an RL loop.

dialogue policy component and achieved better overall performance. Research on DST have been actively investigated, as mentioned in Section 2.3

The main focus of this thesis is concerned with learning a dialogue policy: a function $\pi(\mathbf{b}(s)) = a$ that determines which action to take given a belief distribution $\mathbf{b}(s)$. This is further discussed in detail in the next section.

3.1.2 Overview of Dialogue Policy Optimisation Methods

The spoken dialogues considered here have a finite number of steps T [1]. This type of finite time-step (horizon) RL task is called an episodic task. The dialogue policy π is trained to maximise the episodic cumulative turn-based reward over the entire dialogue:

$$R_t^{\pi} = \sum_{i=0}^{T-t-1} \gamma^t r_{t+i+1} \tag{3.2}$$

To determine an optimal policy π^*, two categories of methods are usually adopted: *value-based* and *policy-based* methods. The relationship between these two is shown in Figure 3.3, where the intersection is often referred to as the *Actor-Critic* method. Details of each approach are discussed in the following sections.

For tasks such as SDS which assume the input utterance from the user is grounded by finite and discrete semantics (states), a POMDP with finite state is

[1]Typically in the range of 2 to 20

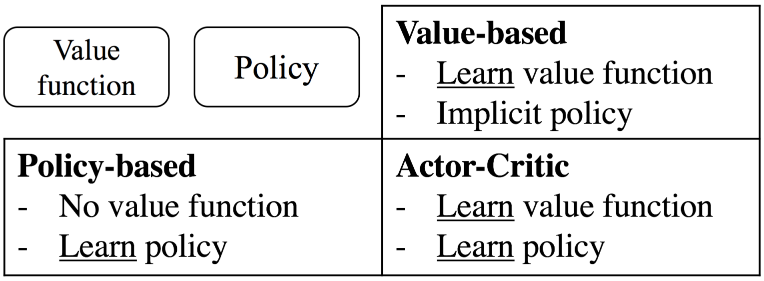


Fig. 3.3 An overview of RL training methods.

adopted. A discrete-state POMDP can be cast as a continuous-state MDP. The following policy optimisation methods are mostly explained in the context of MDPs since they can be easily extended to POMDPs.

3.1.3 Value-based Methods

For MDPs, when using value-based approaches, the expected accumulated reward R_t^π at state $s \in S$ following the policy π is often modelled by the *value function* $V^\pi(s) : S \to \mathbb{R}$:

$$V^\pi(s) = \mathbb{E}_\pi(R_t^\pi | s_t = s) = \mathbb{E}_\pi\Big(\sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1} | s_t = s\Big), \tag{3.3}$$

where the expectation $\mathbb{E}_\pi$ is computed over all possible state sequences generated by the policy π.

Similarly, the value function can be defined not only on states, but also on actions. The *Q-function* $Q^\pi : S \times A \to \mathbb{R}$ is thus defined as the expected accumulated reward at the given state-action pair (s, a) following the policy π:

$$Q^\pi(s,a) = \mathbb{E}_\pi(R_t^\pi | s_t = s, a_t = a) = \mathbb{E}_\pi\Big(\sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1} | s_t = s, a_t = a\Big), \tag{3.4}$$

where, likewise, the expectation $\mathbb{E}_\pi$ is calculated over all possible state-action sequences generated by the policy π.

The relation between the value function $V^{\pi}(s)$ and the Q-function $Q^{\pi}(s,a)$ is thus:

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s)). \tag{3.5}$$

Given a finite state space S, the exact solution to the optimal value function satisfies the *Bellman optimality equation* $V^* = BV^*$ (Bellman, 1954):

$$V^*(s) = \max_a \sum_{s' \in \mathcal{S}} P(s,a,s')(R(s,a) + \gamma V^*(s')). \tag{3.6}$$

In a similar fashion, the optimal Q-function is expressed by the following:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s,a,s')(R(s,a) + \gamma \max_{a'} Q^*(s',a')). \tag{3.7}$$

The optimal Q-function $Q^*(s,a)$ and the optimal value function $V^*(s)$ are related by the following:

$$V^*(s) = \max_a Q^*(s,a). \tag{3.8}$$

Consequently, the optimal policy π^* can be implicitly derived either from the optimal value function:

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s,a,s')(R(a,s) + \gamma V^*(s')), \tag{3.9}$$

or from the optimal Q-function:

$$\pi^*(s) = \arg\max_a Q^*(s,a), \tag{3.10}$$

by choosing an action that maximises the value function or the Q-function.

Solving the POMDP is much harder since the true state if not observable. It consists of determining the best action a to be taken at each belief state b called a *policy* $\pi : B \rightarrow A$, which maximises the expected total discounted reward starting from the belief b. Nevertheless, even if the true state is not known, the optimal value function for each state can be still computed using the transition function $P(o, s_t, a)$ and the observations $o \in \Omega$ seen (Kaelbling et al., 1998) as follows:

$$V^*(s) = \max_a \sum_{s' \in \mathcal{S}} P(s,a,s') \Big(R(s,a) + \sum_{o \in \Omega} P(o,s,a) \gamma V^*(s') \Big) \tag{3.11}$$

The optimal Q-function for POMDPs can thus be computed as follows:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s,a,s') \Big(R(s,a) + \max_{a'} \sum_{o \in \Omega} P(o,s,a) \gamma Q^*(s',a') \Big) \tag{3.12}$$

Then, the optimal value function for any possible belief state $V(b)$ can be computed as the weighted sum over all states using equations 3.1.3

$$V^*(\mathbf{b}) = \sum_{s \in S} b(s_t = s) V^*(s), \tag{3.13}$$

where $b(s)$ is the value of the belief state for the state s. In a similar way, the optimal Q-function can also be computed as a weighted sum using equation 3.12:

$$Q^*(\mathbf{b},a) = \sum_{s \in S} b(s_t = s) Q^*(s,a). \tag{3.14}$$

However, finding an exact algorithm for POMDP is difficult due to the continuity of the belief state. It is therefore non-trivial to solve a POMDP (Murphy, 2000; Papadimitriou and Tsitsiklis, 1987), especially in a real-world task, and thus requires approximation methods (Buşoniu et al., 2011). Point-based methods (Pineau et al., 2003; Shani et al., 2013) are effective by constraining the planning to relevant reachable belief states when computing the value function, but they are not scalable to large state-action spaces and are often model-based approaches that require the estimation of the environment (transitions). Model-free methods are another family for solving POMDPs that strike the balance between exploring the environment and exploiting the learnt policy and directly updating the value functions without learning the transition models. In this case, sample-efficiency is a deciding factor in whether the model can realistically be employed on-line for live applications. GP-SARSA (Gasic and Young, 2014) and Kalman temporal-difference (Daubigney et al., 2012a) are methods that bootstrap estimates of sparse value functions from minimal numbers of training samples and are thus useful for real-time SDS.

3.1.4 Policy-based Methods

Instead of being implicitly defined by a value function, the policy π can also be directly parametrised. Policy-based methods, or *Policy Search* (Deisenroth et al., 2013), directly operate in the parameter space Θ to find the policy $\pi_\theta(a|s)$ (or $\pi_\theta(a|b)$ in POMDP) with parameters $\theta \in \Theta$ and avoid learning a value function. This is shown in the bottom right of Figure 3.3.

Recall from Equation 3.2 and 3.3 that episodic RL can be viewed as the optimisation problem to maximise the expected reward $J(\pi)$:

$$\pi^* = \max_\pi J(\pi) = \max_\pi \mathbb{E}[R|\pi], \tag{3.15}$$

where R is the total episodic reward collected using policy π. If a parametrised model π_θ is used for the policy, the solution is to maximise the expected reward $J(\theta)$:

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \mathbb{E}[R|\pi_\theta]. \tag{3.16}$$

Similar to value-based methods, this optimisation can also be divided into model-based and model-free approaches. The former fully understands the dynamics (transition probability) of the operating environment while the latter does not. Therefore, model-free approaches are normally required to sample the environment during estimation.

One model-free method to find the optimal solution is a derivative-free optimisation approach which treats the entire process as a black box and uses evolutionary strategies (ES) algorithms for heuristic searching. The core idea of ES is as follows: at each iteration (generation), a set of parameter vectors (current generation) is perturbed (mutated) and then evaluated based on the objective function, the total reward R. The parameter vectors with the highest scores on the objective function are then recombined to constitute the population for the next generation. Popular approaches such as cross entropy method (Rubinstein and Kroese, 2013) and natural evolution strategies (Salimans et al., 2017; Wierstra et al., 2014) have been successfully applied to real-world problems. The main problems with ES approaches are inefficiency and incomprehensibility due to them being agnostic to the given problem. Black box optimisation methods are purely 'guessing' the optimal solution.

Policy-gradient methods, on the other hand, aims to optimise the policy with respect to the total reward by gradient ascent. Gradient-based updates guarantee finding a *local* optimum. Assuming the trajectory τ sampled from policy π_θ, the *policy gradient* of $J(\theta)$ with respect to the parameter θ is derived as follows::

$$\begin{aligned}
\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] &= \nabla_\theta \int_\tau p(\tau|\pi_\theta) R(\tau) d\tau \\
&= \int_\tau \nabla_\theta p(\tau|\pi_\theta) R(\tau) d\tau \\
&= \int_\tau p(\tau|\pi_\theta) \frac{\nabla_\theta p(\tau|\pi_\theta)}{p(\tau|\pi_\theta)} R(\tau) d\tau \\
&= \int_\tau p(\tau|\pi_\theta) [\nabla_\theta \log p(\tau|\pi_\theta) R(\tau)] d\tau \\
&= \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log p(\tau|\pi_\theta) R(\tau)]
\end{aligned}$$

The derivation proceeds from the third to the fourth step by using the *Likelihood ratio trick*: $\nabla \log p(x) = \frac{1}{p(x)} \nabla p(x)$, where $\nabla \log p(x)$ is the *score function*. Using the chain rule in probability theory, we can further expand the trajectory probability $p(\tau|\pi_\theta)$ as follows:

$$p(\tau|\pi_\theta) = \mu(s_0)\pi(a_0|s_0,\theta)p(s_1,r_1|s_0,a_0)\pi(a_1|s_1,\theta)\dots\pi(a_{T-1}|s_{T-1},\theta)p(s_T,r_T|s_{T-1},a_{T-1}),$$

where $\mu(s_0)$ is the initial state distribution. When calculating $\nabla_\theta \log p(\tau|\pi_\theta)$, the sequence product turns into a sum, and the differentiation with respect to θ cancels the terms $\mu(s_0)$ and $p(s_t, r_t|s_{t-1}, a_{t-1})$ to avoid having to know the dynamics (transition probability) of the environment. This leads to the final equation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) R(\tau)\right]. \tag{3.17}$$

Therefore, setting a learning rate α, the update of the policy parameters θ is as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta). \tag{3.18}$$

Considering a single reward at time t, Equation 3.17 can be rewritten as the following:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_t] = \mathbb{E}_{\tau\sim\pi_\theta}\left[r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi(a'_t|s'_t)\right]. \tag{3.19}$$

The (total) reward r_t is obtained when performing a sequence of actions $a_{t'}$, with $0 \leqslant t' \leqslant t$. When summing over the entire trajectory ($\sum_{t=0}^{T-1}$), it can further be presented as follows:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi(a'_t|s'_t)\right] \tag{3.20}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \sum_{t'=t}^{T-1} r'_t\right]. \tag{3.21}$$

The term $\sum_{t'=t}^{T-1} r'_t$ in the above equation is the total reward collected from time-step $t \sim T-1$. It can be further replaced by Q-value function in Equation 3.4:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) Q^{\pi_\theta}(s_t, a_t)\right]. \tag{3.22}$$

This equation is known as *policy gradient theorem* (Sutton et al., 2000).

In practice, the gradient is estimated by a batch of N samples collected using the policy π_θ from the environment, where the accuracy increase when $N \to \infty$. One way to solve this is the REINFORCE algorithm (Monte-Carlo method) (Williams, 1992), which directly samples the total reward following the policy π_θ:

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N}\sum_{n=1}^{N}\Big(\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t)\sum_{t'=t}^{T-1} r'_t\Big) \tag{3.23}$$

Since this form of gradient has a potentially high variance, it can hinder the learning efficiency. Hence, a baseline function $b(s)$ is typically introduced to reduce the variance while not changing the estimated gradient (Sutton and Barto, 1999; Williams, 1992). This baseline can be any arbitrary function. However, the best candidate for this baseline $b(s)$ is the value function $V(s)$ (Greensmith et al., 2004). Eq. 3.23 then becomes the following:

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N}\sum_{n=1}^{N}\Big(\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t)\Big(\sum_{t'=t}^{T-1} r'_t - b(s_t)\Big)\Big) \tag{3.24}$$

The difference $\sum_{t=t'}^{T-1} r_t - b(s_t)$ demonstrates whether the total reward using the current policy is better than *expected* ($V(s_t)$), showing how good the current action a_t is. This can be further cast as the *advantage function* $A_t = \sum_{t'=t}^{T-1} r_{t'} - b(s_t)$, and a detailed description on the advantage function is discussed in Chapter 7.

One can also adopt a separate function, a *critic* with parameters w, to estimate this Q-function $Q^{\pi_\theta}(s_t, a_t)$ in Equation 3.22:

$$Q_w(s_t, a_t) \approx Q^{\pi_\theta}(s_t, a_t). \tag{3.25}$$

This leads to the *actor-critic* algorithm (Grondman et al., 2012; Konda and Tsitsiklis, 1999):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t) Q_w(s_t, a_t)\Big], \tag{3.26}$$

which has two sets of parameters: w for the critic and θ for the actor (policy). The gradient ascent direction of the actor is suggested by the critic. The critic here is basically a Value- or a Q-function; therefore, most of the approaches in the previous section can be utilised. The main advantage of this method is that the policy can be directly modelled, and the critic provides a good estimation of the expected total reward to reduce the variance.

Policy-based RL methods have been successfully applied to various tasks such as robotics (Deisenroth et al., 2013; Peters and Schaal, 2006), games (Silver et al., 2016), and non-differentiable computations in vision and language tasks (Mnih et al., 2014; Yu et al., 2017). Its application to SDS are also numerous (Dhingra et al., 2016; Fatemi et al., 2016; Jurčíček, 2012; Li et al., 2016; Williams et al., 2017). The main benefits of such a method are the following: 1) it can learn stochastic policies, 2) it is effective in high-dimensional or continuous action spaces (Schulman et al., 2016), and 3) it has better convergence properties compared to value-based methods, since the policy is directly modelled and optimised towards the desired objective. More details are provided in Chapter 7.

3.1.5 Taxonomy of Reinforcement Learning Approaches

In the previous sections, two main methods for finding an optimal policy have been introduced. As mentioned above, policy-based approaches have stronger convergence characteristics than value-based methods. The latter often diverge when using function approximation since they optimise in value space and a slight change in value estimate can lead to a large change in policy space (Sutton et al., 2000). Policy-based methods, especially policy-gradient approaches, suffer from low sample-efficiency, high variance and often converge to local optima, since they typically update using gradient-based methods. In this dissertation, both training methods will be utilised and explored. For the first part of the thesis (Chapters 4 – 6), GP-SARSA (Gasic and Young, 2014), a value-based method, is used for policy learning. For the second part (Chapter 7), a neural network model is used to parametrise the policy and is trained with policy-based and actor-critic methods.

There are a variety of different dimensions which characterise various RL approaches, and these are presented in the following sections.

Episodic and Continuing task

As described above, the goal is to find a policy π that maximises the expected discounted reward R_t^π starting at time-step t. There are two types of RL problems: 1) the learning process is divided into several finite-length episodes, and 2) the learning process has only a single episode which continues indefinitely. The first case is called the *episodic task*, and the second case is referred to as the *continuing task*. In episodic tasks, an initial and a terminal state are clearly defined. The agent restarts in the initial state for each new episode after the previous one ends. As described in section 3.1.2, SDS is an example of an episodic task which has a finite number of interactions. Other examples such as games and mazes are also episodic. For continuing tasks, the formal expected discounted reward is defined as in Equation 3.1. In this case, the discount factor γ is normally set to less than 1 to prevent infinite total reward.

Model-based and Model-free

The difference between *model-based* and *model-free* methods is whether the knowledge (dynamics) of the environment is known. In the POMDP setting, this knowledge includes the transition and observation functions. If this information is available, model-based dynamic programming algorithms such as value iteration or policy iteration (Sutton and Barto, 1999) can be adapted to obtain an exact solution. These algorithms exploit the recursive characteristics of the Bellman equation in Equation 3.6 to find the optimal value function estimates between every consecutive time-step. Otherwise, this information can also be learnt from data (Kaelbling et al., 1998), and the dynamic programming algorithms can then be used. However, model parameters are often difficult to learn, and learning using a bad model can harm the performance.

Model-free methods, in contrast, do not make any assumption on the POMDP structure and then model the expected total reward directly from the collected rewards. This enables the agent to discover the underlying structure of its operating environment. In the SDS context, model-based approaches are often intractable (Williams and Young, 2007a). Therefore, the main focus of research to date has been on applying model-free methods (Jurčíček, 2012; Young et al., 2013b).

Exploration and Exploitation

When using model-free RL methods, it is often unclear what sort of action the agent should take. This problem is often referred to as the *exploitation/exploration* dilemma: 1) **exploitation**: selecting the optimal action based on the current policy 2) **exploration**: taking a non-optimal action given the current policy to get more information about the environment to estimate the truly optimal policy better. Gathering enough information is always crucial to making the best overall decisions. However, too much exploration may prevent the agent from converging to the optimal solution.

One widely-adopted strategy is the *ϵ-greedy* method, which selects the optimal action based on the current policy with probability $1-\epsilon$, or a random action to explore the environment with probability ϵ:

$$\pi^{\epsilon}(s) = \begin{cases} \arg\max_{a \in \mathcal{A}} \pi(s,a) \text{ with prob. } (1-\epsilon) \\ \text{random } a \in \mathcal{A} \text{ with prob. } \epsilon \end{cases} \tag{3.27}$$

The problem with this method is that all actions are treated equally when exploring. In some cases, certain actions are clearly more informative than others for finding a globally optimal policy. Hence, using this method might take longer to converge to the desired solution. Other approaches such as Boltzmann exploration or Bayesian methods (Gašić and Young, 2014; Ghavamzadeh et al., 2015; Tegho et al.,

2017) exploit the probabilistic estimate of the current policy or its corresponding uncertainty measurement to further improve the convergence rate of policy learning.

Dynamic Programming, Monte-Carlo and Temporal Difference Backup

In RL, the term *backup* represents the process of updating the estimated value of a state using values of future states in an episode. If this value of the future state is also an estimate, this is called the *bootstrapping*. Figure 3.4 demonstrates the backup diagrams of different methods: dynamic programming, exhaustive search, Monte-Carlo (MC) and temporal difference (TD). Circles represent the states, dots represent the executed actions, and shaded squares represent terminal states. Since exhaustive search is usually intractable, it is not discussed in detail here.

Dynamic programming is a model-based approach which bootstraps the current state value based on the values of all possible states in the next time-step rather than one sampled state. On the other hand, MC and TD are model-free methods where the backup is done on the sampled experiences. The difference between MC and TD is whether bootstrapping is adopted. MC must learn from the complete sequence and looks ahead till the end of the episode, where the total reward can be empirically computed and used for updating. It has high variance and zero bias, but it requires more data to converge to a desired policy. TD, in contrast, learns from incomplete sequences and looks ahead only one step. It utilises the estimated value of a state in the next time-step to update the estimate of the current state. This method has low variance and high bias, which depends highly on how good the value estimations are. To combine the best of both, Sutton and Barto (1999) introduced TD(λ), where $\lambda \in [0,1]$ and is related to n-step look ahead total reward. TD(0) is the standard TD method, and TD(1) is equivalent to Monte-Carlo (MC).

On-line and Off-line

On-line RL methods typically update the agent's policy based on each sample (or experience) incrementally in the interaction. The MC approach and TD algorithms

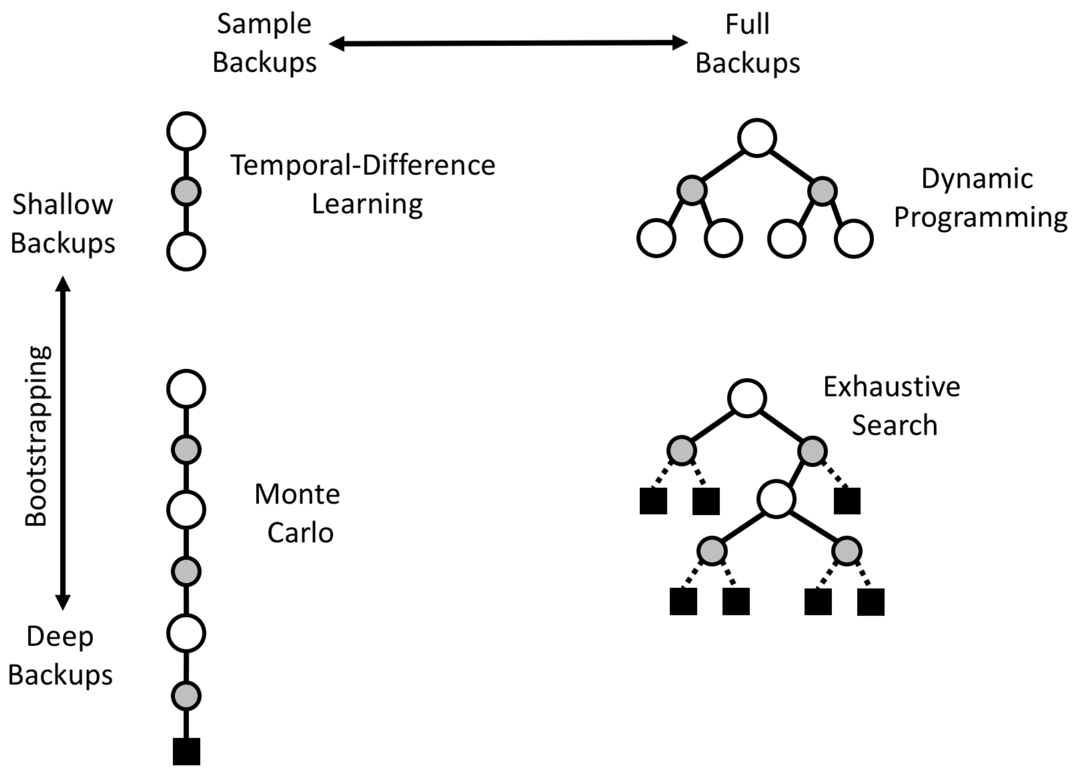


Fig. 3.4 The backup diagrams of different methods redrawn from Sutton and Barto (1999): Dynamic Programming, Exhaustive Search, Monte-Carlo and Temporal Difference. Circles represent the states, dots represent the executed actions, and shaded squares represent the terminal states. The x-axis and y-axis illustrate the width (sample/full) and depth (shallow/deep) of the update.

such as SARSA and Q-learning (Sutton and Barto, 1999) fall into this category. On the other hand, *off-line* (or *batch*) RL approaches update the agent's policy with a batch of samples, which is more sample-efficient since more information is provided in one single parameter update. Fitted-Q Iteration (Antos et al., 2008; Riedmiller, 2005) is one of these applicable methods for real-world tasks.

On-policy and Off-policy

Algorithms to learn the RL policy can also be split into *off-policy* and *on-policy* methods. The policy used to generate training dialogues (episodes) is referred to as the *behaviour policy* μ. This is in contrast to the policy to be optimised which is called the *target policy* π. Finding the target policy π is the goal of all RL problems. When learning with on-policy methods, it is assumed that actions are drawn from the same policy as the target to be optimised ($\mu = \pi$). When using off-policy methods, since the current policy π is updated with a batch of samples generated from some old behaviour policies μ, an importance sampling (IS) ratio is used to rescale each sampled reward to correct the sampling bias at time-step t: $\rho_t = \pi(a_t|\mathbf{b_t})/\mu(a_t|\mathbf{b_t})$ (Meuleau et al., 2000). These old behaviour policies might be collected from the earlier phase of the RL agent or from a corpus of dialogue interactions (Daubigney et al., 2012b). Further investigation and more details on these methods are discussed in Chapter 6.

3.2 Reward Estimation in Dialogues

3.2.1 Dialogue Evaluation

Evaluating an SDS is difficult due to the complexity of the long-term interactions between the user and the system. Unlike most speech and natural language processing tasks where the evaluation metrics are more easily specicfied (Goldstein et al., 1999; Hirschman and Thompson, 1997; Mangu et al., 2000), the definition of a good SDS is very vague. The entire SDS can be broken down into various modules such as SLU and DST, and their performance can be assessed individually. However, the joint evaluation of the whole system is still challenging.

The most natural way to evaluate the dialogue is via human judgement. However, it is often costly to ask a human to evaluate every dialogue, and anyway different users might have their own subjective view of dialogue quality. Thus, an automatic evaluation metric is desired to avoid the costly human-rating process.

A corpus of dialogue data can be collected and treated as a supervised learning task (Bordes et al., 2017; Vinyals and Le, 2015). The goal of the model there is to 'mimic' the responses collected in the data and be evaluated by similarity metrics such as BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) on the generated natural language, which are widely used in the machine translation literature. Nonetheless, it has been shown that these *word-overlap similarity metrics* have low correlation with human ratings (Liu et al., 2016). Furthermore, the supervised training sets often lack sufficient coverage of all the possible dialogue scenarios.

Since task-oriented SDS is the focus of this thesis, task-completion is a straightforward automatic measurement which checks whether the information the system provides matches the user goal. As mentioned in Section 2.4, the quality of a dialogue is often related to the performance of the dialogue policy, which controls the flow of the interaction. In the SDS pipeline, this task completion information can serve as the high-level learning objective of the dialogue policy. It is often referred to as the *reward* in an RL setting. However, it is hard to measure due to the unknown objective of the user when assessing a dialogue task on-line. Since a robust reward metric is critical for building an SDS, it is studied in depth in this thesis.

3.2.2 Heuristic Dialogue Reward Settings

A clear objective is the key to success and the reward function plays an important role in the POMDP framework and in RL tasks generally. It defines the desired behaviour of the learning agent (Dewey, 2014). Typically, a positive reward is assigned when certain situations related to task success are reached and vice versa. For instance, in robotics control and planning problems, chess, and Atari games, the common characteristic between these tasks is that they often have well-defined evaluations and clear goals. However, as mentioned above, the term 'success' is not always well-defined in SDS applications, since they involve the feedback of human users that are often unknown to the system. Moreover, in SDS, per-turn feedback is costly and often infeasible; thus, only a sparse reinforcement signal is obtained as a result. In several task-oriented examples to date (Casanueva et al., 2015; Lemon et al., 2006; Williams and Young, 2007b), the definition of success is often formulated based on a predefined task given to the users and determined by whether all slots (requirements) described in the task are fulfilled. In addition, a per turn penalty is applied for shortening the dialogue.

The reward is highly related to the 'goodness' of the interaction in SDS, which is often determined by the user satisfaction. The way it is defined also dramatically affects the learning speed of the dialogue policy. This work presents a review of different perspectives to reward modelling.

3.2.3 The PARADISE Evaluation Framework

The PARADISE (PARAdigm for DIalogue System Evaluation) is a unified evaluation framework (Walker et al., 1997) that defines the user satisfaction of the SDS performance as the weighted sum of the dialogue success (task completion), the predefined dialogue costs such as dialogue duration, and the counts of confirmations the system has made. Multivariate linear regression is adopted to estimate the weights given a labelled dialogue corpus. A comparison across domains is made possible due to normalisation of the measurement, and the contribution of each dialogue cost can be found from the learnt weights. This measure was later used as a reward function for learning a dialogue policy (Rieser and Lemon, 2011). However, as noted, task completion is rarely available when the system is interacting with real users, and concerns have been raised regarding the theoretical validity of the model (Larsen, 2003).

3.2.4 Inverse Reinforcement Learning

Quantifying user satisfaction is non-trivial. Rather than explicitly defining a reward function, inverse reinforcement learning (IRL) aims to recover the underlying reward from demonstrations of good behaviour and then learn a policy which maximises the recovered reward (Russell, 1998). IRL was first introduced to SDS in Paek and Pieraccini (2008), where the reward was inferred from human-human dialogues to mimic the behaviour observed in a corpus. IRL has also been studied in a Wizard-of-Oz (WoZ) setting (Boularias et al., 2010; Rojas Barahona and Cerisara, 2014), where typically a human expert served as the dialogue manager to select each system reply based on the speech understanding output at different noise levels. However, this approach is costly and there is no reason to suppose that a human wizard is acting optimally, especially at high noise levels. IRL has also been introduced to various aspects of SDS, such as dialogue management (Kim et al., 2014; Sugiyama et al., 2012) and user simulation (Chandramohan et al., 2011). An overview of IRL for SDS is summarised in (Pietquin, 2013).

3.2.5 Coherence in Dialogue Systems

Another stream of dialogue evaluation research that is potentially useful for reward design is naturalness assessment of the system reply, an example of which is the coherence of the generated dialogue responses to the user utterances (Gandhe and Traum, 2008; Higashinaka et al., 2014) that encode lexical and semantic information. When employing this in SDS, the main focus of the policy learning is the semantic features (whether the system replies reasonably), while the lexical representation is less focussed; as it is realised in the NLG component after the decision (semantics) is made in the dialogue manager.

3.2.6 Reward Shaping

Unlike supervised learning, instructive feedback is not provided for each input in RL learning systems. This leads to the *temporal credit assignment problem* (Sutton, 1984), where the reward signals are usually weak and delayed. Therefore, learning is often slow.

A straightforward way to address this issue is to provide the system with an extra reward signal F in addition to the original environmental reward R. This makes the system learn from the composite signal $\tilde{R} = R + F$. This reward F often encodes expert knowledge that complements the sparse signal R (Mataric, 1994).

Since the reward function defines the objective of the system, modifying it may also change the original problem into a new problem. In this case, a poorly-designed reward function might degrade learning, learnt policies may never reach the goal of the original problem (Randløv and Alstrøm, 1998).

Consequently, Ng et al. (1999) proposed potential-based reward shaping (PBRS). When the task is modelled as a fully observable MDP, formal requirements on shaping reward are defined as a difference between potential functions ϕ on consecutive states s and s'. This is guaranteed to preserve the optimality of policies:

$$F(s_t, a, s_{t+1}) = \gamma\phi(s_{t+1}) - \phi(s_t), \tag{3.28}$$

where γ is exactly the same discount factor used for policy learning. The potential function ϕ enables domain or expert knowledge to be incorporated. Thus the resulting expected total reward will be as follows:

$$\begin{aligned}
&\mathbb{E}[\tilde{r}_0 + \gamma\tilde{r}_1 + \gamma^2\tilde{r}_2 + ...] \\
&= \mathbb{E}[(r_0 + \gamma\phi(s_1) - \phi(s_0)) + \gamma(r_1 + \gamma\phi(s_2) - \phi(s_1)) + \gamma^2(r_2 + \gamma\phi(s_3) - \phi(s_2)) + ...] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + ... - \phi(s_0)].
\end{aligned}$$

Thus, the corresponding value function, Q-value function, and advantage function are as follows:

$$\begin{aligned}
\tilde{Q}^\pi(s,a) &= Q^\pi(s,a) - \phi(s) \\
\tilde{V}^\pi(s) &= V^\pi(s) - \phi(s) \\
\tilde{A}^\pi(s,a) &= (Q^\pi(s,a) - \phi(s)) - (V^\pi(s) - \phi(s)) = A^\pi(s,a).
\end{aligned}$$

Based on these equations, the optimal policy, either using the value-based method or using the policy-based method, is invariant to the shaping reward

(Schulman et al., 2016). Given this property, its extension to POMDPs was shown to hold by proof and empirical experiments (Eck et al., 2015):

$$F(b_t, a, b_{t+1}) = \gamma\phi(b_{t+1}) - \phi(b_t), \tag{3.29}$$

where b_t represents the belief state at turn t, and a represents the action leading b_t to b_{t+1}.

However, determining an appropriate potential function for an SDS is non-trivial. Ferreira and Lefèvre (2013) have proposed the use of reward shaping via analogy to the 'social signals' naturally produced and interpreted throughout a human-human dialogue. This non-statistical reward shaping model used heuristic features for speeding up policy learning. As an alternative, one may consider attempting to handcraft a finer grained environmental reward function. For example, Asri et al. (2014) proposed diffusing expert ratings of dialogues to the state transition level to produce a richer reward function. Policy convergence may occur faster in this altered POMDP and dialogues generated by a task based simulated user may also alleviate the need for expert ratings. However, unlike PBRS, modifying the environmental reward function also modifies the resulting optimal policy.

Determining an appropriate potential function for an SDS is non-trivial. Rather than handcrafting the function with heuristic knowledge, Chapter 6 presents a statistical method for predicting proper values for shaping rewards.

3.3 Summary

RL has been shown to be effective in learning a dialogue policy. This chapter has described two crucial factors: the algorithms for policy learning and the reward estimation methods for guiding a better policy. In the first half of the chapter, the basics and training methods for RL-based policy were explained. The design and estimation of the reward function for dialogue applications was then investigated in the second half. As noted, computing a reward function in a real-world SDS is not straightforward. Hence, the next two chapters explore this issue further and present some new approaches.

CHAPTER 4

Policy Learning with Off-line Reward Estimator

4.1 Motivation

The ability to compute an accurate reward function is essential for optimising a dialogue policy via RL. However, evaluating a dialogue on-line in order to compute a reward for training the policy is difficult since real users have their own goal in mind which is not available to the system. In most previous works, the training of an SDS has been done with either recruited subjects (Gašić et al., 2014; Gašić and Young, 2014) presented with a pre-defined task to complete, or via simulated users (Daubigney et al., 2012a; Keizer et al., 2010b; Lemon and Pietquin, 2007; Levin et al., 2000; Schatzmann and Young, 2009) who randomly sample a goal over an ontology. In both cases, the specific prior knowledge of the user's goal is used to calculate an objective measure (*Obj*) of whether the SDS completed the task. In these systems, a common issue is that users are not motivated by a real information need. THus, they often[1] fail to follow exactly the presented goal, resulting in *Obj*=failure even though the SDS may have provided everything asked for it. To avoid penalising the SDS by learning from such inaccurate reward signal, the user was also asked for their opinion on whether they achieved the task goals, thereby obtaining a subjective success rating (*Subj*). For policy learning, only dialogues for which *Obj*=*Subj* (Gašić et al., 2013b) are used, and the remainder are discarded.

[1]This case occurs in our experience at least 20% of the time (Gašić et al., 2011).

In real-world systems, this prior knowledge of the user's goal is simply not available, making any calculation of an 'objective' measure nearly impossible. However, knowledge of task success or failure is essential for training an SDS. Of course in some situations it is possible and even natural for the system to ask the user for feedback at the dialogues conclusion (e.g. confirming a purchase with a *yes/no* question); however, this is not always possible and also user responses can be 'noisy' (Larsen, 2003), which results in slower learning. It is, therefore, essential to find effective methods for computing rewards with real users when the underlying task is unknown.

Here, a reward estimator is presented which can be trained to estimate task success. This reward estimator is realised by an RNN, which is sequentially inputted with the information of the human-machine interaction that appears at each turn. It requires training data annotated for task success which in the work described here is generated off-line by a user simulator. Once trained, the reward estimator enables on-line policy optimisation with real users without access to either *Obj* or *Subj* task success ratings.

4.2 Recurrent Neural Network Reward Estimator

RNNs are a subclass of neural network models with recurrent connections from one time-step to the next. Their ability to succinctly capture and retain history information makes them suitable for modelling sequential data with temporal dependencies. They have been previously used with success in a variety of natural language processing (NLP) tasks such as language modelling (Karpathy and Fei-Fei, 2014; Mikolov et al., 2010, 2011b), SLU (Mesnil et al., 2015b) and NLG (Wen et al., 2015).

The core idea of the RNN is the information flow between different time-steps within the model. A commonly used RNN, Elman-type RNN (Elman, 1990), is depicted in Figure 4.1. This model includes a hidden layer h_t at time-step t that

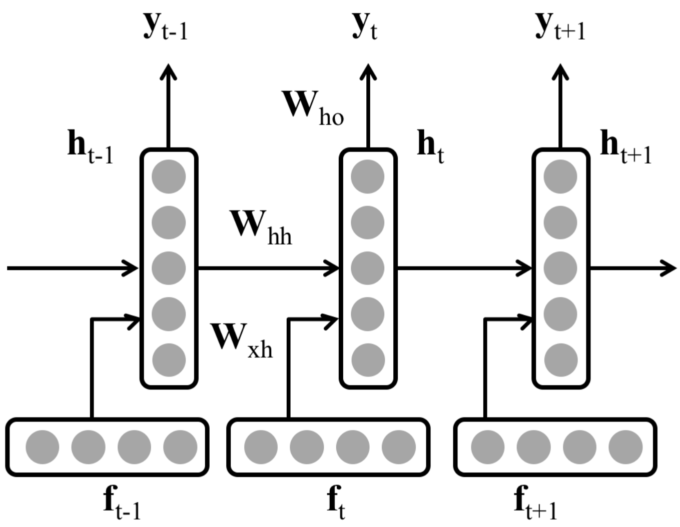


Fig. 4.1 Recurrent neural network architecture with three time-steps.

performs the addition of linear transformations of the input x_t and the output of the previous step, and then it uses a non-linear operation for predicting the output $\hat{y}_t$. The mathematical representation is as follows:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1}) \tag{4.1}$$

$$\hat{y}_t = g(W_{ho}h_t), \tag{4.2}$$

where the Ws are the weight matrices between different layers, $f(\cdot)$ is a non-linear activation such as a `sigmoid` or `tanh` and $g(\cdot)$ is another transformation from real-valued h_t to the desired output such as a `softmax` for an output probability distribution. Additional learnable weights b can also be added to bias the output in the hidden and output layers.

A loss function is defined to calculate the error between the prediction and true label for performing gradient-based methods such as back-propagation through time (BPTT) to update the weights. However, basic RNN models often suffer from vanishing or exploding gradient problems that limit their capability of modelling

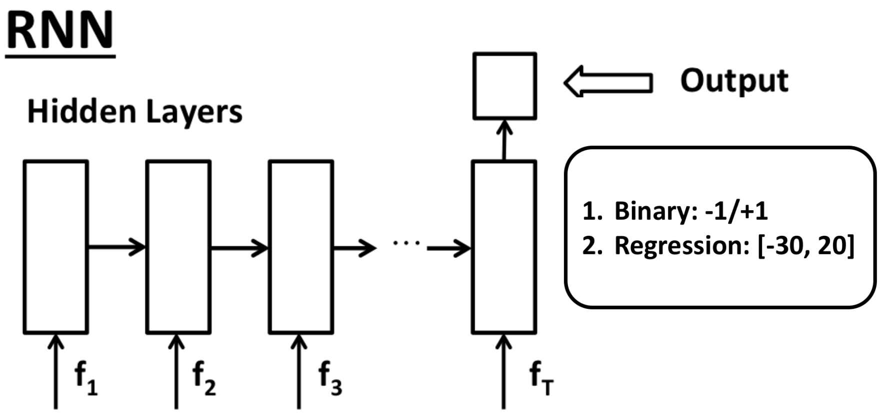


Fig. 4.2 An unrolled view of the RNN reward estimator. The feature vectors extracted at turns $t = 1,\ldots,T$ are labelled $\mathbf{f}_t$. Two types of output are considered: binary classification and regression.

long context dependencies due to the multiple derivations of the loss function for gradient calculation as described in Bengio et al. (1994). To address such issues, several methods have been exploited, such as *gradient clipping* (Mikolov et al., 2011a) that avoids aberrant changes in parameters and rectified linear unit (ReLU) activations that restrict the gradient of the nonlinearity to be 0 or 1. Alternatively, sophisticated mechanisms within the hidden layers such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) have also been proposed. More details on these advanced gating mechanisms are discussed in Chapter 6.

The RNN reward predictor takes as input the feature vectors $\mathbf{f}_t$ described above and updates its hidden layer $\mathbf{h}_t$ at each turn t.[2] Once the dialogue ends, the hidden layer is then connected to an output layer to make a single reward prediction for the whole dialogue, as depicted in Figure 4.2.

The network structure in the final layer is determined by the choice of supervised training targets, of which two types were considered:

[2] Convolutional neural networks were also investigated on the same task as in Su et al. (2015a) However, only results using an RNN are shown here since it was found to be generally more robust.

1) Classification model: in this case, the RNN is formulated as a binary classifier, which is trained to predict success or failure for each dialogue. The targets are $\{0,1\}$, and the final layer of the RNN outputs a scalar through a sigmoid activation function and is trained with a cross-entropy loss. The output from this network is a probability p that the dialogue is a success, and the hard class label predicted by the model is taken as 1 if $p > 0.5$, else 0. This hard label is used to determine whether to give a final reward of +20 during policy learning, as per Equation 4.3.

2) Regression model: in this case, the RNN is formulated as a regressor with the actual return value used as the training target. The model's final layer has no non-linearity (activation) and the whole model is trained with a mean-square-error (MSE) loss function. During subsequent policy learning using this predictor, the per-turn penalty is suppressed since it is already considered by the RNN reward predictor.

4.3 Training data and dialogue features

Building the proposed RNN reward prediction model requires a set of training data, the human-machine dialogues and the corresponding task success annotations, which was collected by training several GP-SARSA policies (Gašić and Young, 2014) under various error rates (see Section 4.4 below) from scratch using an agenda-based simulated user (Keizer et al., 2010b; Schatzmann et al., 2006). Each dialogue was labelled as a success or failure using the objective criteria (*Obj*) described in the introduction, that is, a dialogue was labelled as successful if all of the users' goals randomly generated at the start of the dialogue were completed. The reward function used during policy training gave -1 at each turn to encourage brevity and +20 at completion if the dialogue was successful, otherwise 0. The return (cumulative reward) $\Re$ was, therefore, calculated as follows:

$$\Re = 20 \times \mathbb{1}(\mathcal{D}) - N \tag{4.3}$$

where N is the number of dialogue turns and $\mathbb{1}(\mathcal{D})$ is an indicator function for success.

For all models, a feature vector was extracted at each turn consisting of the following concatenated sections, which is also shown in Figure 4.3:

- a one-hot encoding of the user's top-ranked dialogue act
- the real-valued belief state vector formed by concatenating the distributions over all goals (area, price range, food type), methods and history variables (Thomson and Young, 2010)
- a one-hot encoding of the summary system action
- the current turn number

Fig. 4.3 Feature vector $\mathbf{f}_t$ extracted at each turn t.

This form of feature vector was motivated by considering the primary information a human would need to read in a transcription in order to rate the success of the dialogue. The inclusion of the belief vector, the user dialogue act and the system's actions make this feature vector domain and system dependent.

The aim of the reward predictor is to enable policy learning with real users without prior knowledge of their goals. To do this, the model should consider the information available at every dialogue turn and then evaluate whether the policy provided everything that was requested. The hope is that by training the reward model on data from simulated users, it will generalise and provide accurate assessments of dialogues with real users whose goals are not known *a priori*.

4.4 Experiments

The Cambridge restaurant domain is used for all the experimental work described in this thesis. Users converse with the system to find restaurants in Cambridge that match their required constraints such as food type and area. This Cambridge restaurant domain consists of approximately 150 venues and each venue has six attributes (slots) of which three (area, price range and food type) can be used by the system to constrain the search and the remaining three (phone number, address and postcode) are informable properties that can be queried by the user once a database entity has been found.

4.4.1 The Core Spoken Dialogue Systems

Two operating modes are used in the experiments, **live user trial mode** and **user simulation mode**.

- In live user trial mode, the core components of the SDS comprise a domain-independent HMM-based speech recogniser, a confusion network (CNet) semantic decoder (Henderson et al., 2012), the BUDS belief state tracker that factorises the dialogue state using a DBN (Thomson and Young, 2010), a GP-based dialogue policy (Gašić and Young, 2014), and a template-based natural language generator (NLG) to map system actions into natural language responses back to the user.

- In user simulation mode, an agenda-based simulated user is used to interact with the system at the abstract dialogue act level (Keizer et al., 2010b; Schatzmann et al., 2006). In this mode, the system consists of only the BUDS belief state tracker and the dialogue policy. The user simulator includes an error generator (Schatzmann et al., 2007b), which can be set to generate user inputs with different error rates at the semantic level, namely semantic error rates (SER).

The dialogue policy π maps the *belief state* $\mathbf{b} \in \mathcal{B}$, a distribution over all possible dialogue states at each turn, to a system action $a \in \mathcal{A}$, which in live mode is converted into the natural language using the NLG component.

As explained in Chapter 3, the behaviour of an SDS is conditioned by a reward function $r(\mathbf{b},a)$ which for any given belief state $\mathbf{b}$ and action a defines the *immediate reward* for that turn. Dialogue policy optimisation seeks to use RL to maximise the total reward expected over each complete dialogue.

The Q-function as introduced in Equation 3.4 estimates the expected cumulative reward [3] (also called the *return*) from any given belief state ($\mathbf{b}$ and action a to the end of the dialogue. Optimising the Q-function is equivalent to optimising the policy π.

For all of the experiments described in this chapter, the Q-function is modelled by a GP:

$$Q(\mathbf{b},a) \sim \mathcal{GP}\left(m(\mathbf{b},a),k((\mathbf{b},a),(\mathbf{b},a))\right), \tag{4.4}$$

where $m(\cdot,\cdot)$ is the prior mean function and the kernel $k(\cdot,\cdot)$ is factored into separate kernels over belief and action spaces $k_{\mathcal{B}}(\mathbf{b},\mathbf{b}')k_{\mathcal{A}}(a,a')$. The policy is optimised using GP-SARSA (Engel, 2005; Gašić and Young, 2014) in which the Q-function is updated by calculating the posterior given the collected belief-action pairs $(\mathbf{b},a)$ (dictionary points) and their corresponding rewards.

GP-based RL (GPRL) is particularly appealing since it can learn from a small number of observations by exploiting the correlations defined by the kernel function and, as a bonus, it provides a measure of the uncertainty in its estimates. This knowledge of the distance between data points in the observation space greatly speeds up policy learning since the Q-values of the unexplored space can be estimated from the Q-values of nearby points. However, computation and model complexity become intractable if every data point must be memorised. So instead,

[3] The total reward in RL often includes a discount factor to discount the value of future rewards. Dialogues in a task-oriented SDS typically require only 6 or 7 turns, and they rarely extend beyond 20 turns. Hence the discount factor is normally set to 0.99 or 1.0.

sparse approximation methods such as *kernel span* (Engel, 2005) are used to reduce the size of stored training points.

Further reductions in data space requirements can be achieved by condensing the action set into a fixed set of *summary* actions, which are then heuristically mapped back into the *master* system action space using a set of hand-crafted rules (Gašic, 2011). The summary action space consists of a set of slot-dependent and slot-independent summary actions. Slot-dependent summary actions include requesting the slot value, confirming the most likely slot value and selecting between the top two slot values.

The summary action kernel is defined as $k_{\mathcal{A}}(a, a') = \delta_a(a')$, where $\delta_a(a') = 1$ if $a = a'$ and 0 otherwise. The belief state consists of the probability distributions over the Bayesian network hidden nodes that relate to the dialogue history for each slot and the user goal for each slot. The dialogue history nodes can take a fixed number of values, whereas user goals range over the values defined for that particular slot in the ontology and can have very high cardinalities. User goal distributions are therefore sorted according to the probability assigned to each value since the choice of summary action does not depend on the values but rather on the overall shape of each distribution. The kernel function over both dialogue history and user goal nodes is based on the expected likelihood kernel Jebara et al. (2004), which is a simple linear inner product. The kernel function for belief space is then the sum of all the individual hidden node kernels:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}') = \sum_{\mathbf{h}} \langle \mathbf{b_h}, \mathbf{b'_h} \rangle, \tag{4.5}$$

where $\mathbf{b_h}$ is the probability distribution encoded in the hth hidden node.

4.4.2 RNN-based Dialogue Success Prediction

The off-line reward estimator described in the previous section was implemented using the Theano library (Bastien et al., 2012; Bergstra et al., 2010) with a hidden layer of 300 units and sigmoid activation units. All models were trained using SGD (per dialogue) on data generated by three independently trained GP-based policies interacting with a simulated user at a 15% semantic error rate (SER) (Schatzmann and Young, 2009). One model was fully trained on a set of 18K dialogues (`train-18K`), and the second model was trained on a smaller set of only 1K dialogues (`train-1K`). The contrast between the larger `train-18K` set and the smaller `train-1K` set is provided to give some insight into the impact of reduced availability of training data. A separate validation set of 1K dialogues was also generated to avoid overfitting.

There were two test sets: a matched set of 1K dialogues generated at SER 15% (`testA`) and a larger set containing 3K dialogues at each of four SERs: 0%, 15%, 30%, and 45% (`testB`), where the latter provides an indication of how models perform when there is a mismatch between train and test sets.

For each dialogue, the input features were represented as the concatenation of four segments as shown in Figure 4.3, which were realised in two configurations, a domain-dependent feature vector $\mathcal{F}_{617}$ and a domain-independent feature vector $\mathcal{F}_{74}$ (Vandyke et al., 2015), where the subscripts denote the size of the vectors. For $\mathcal{F}_{617}$, the number of elements were 21, 575, 20, and 1, respectively, for the most likely user act, the full belief state (distributions over all values for each informable slot: area, price range, food type and binary predictions on requestable slots), the one-hot encodings for each of the system actions and the turn number normalised by the maximum number of turns, which is 30 in this configuration. The one-hot user dialogue act encoding was formed by taking only the most likely user act estimated by the CNet semantic decoder. The user acts are described in Table 4.1.

Table 4.1 List of user dialogue acts in the F_{617} set.

hello	inform	request	select	confirm
affirm	negate	deny	silence	thankyou
ack	bye	hangup	reqalts	reqmore
confreq	help	repeat	restart	null
canthear				

The corresponding detailed descriptions of each user dialogue acts are illustrated in A. Table 4.2 shows the list of the summary system actions.

Table 4.2 The list of summary system actions.

System Action	Description
`request_area`	system asks user to provide the value of area
`request_pricerange`	system asks user to provide the value of price range
`request_food`	system asks user to provide the value of food type
`confirm_area`	system asks user to confirm the value of area
`confirm_pricerange`	system asks user to confirm the value of price range
`confirm_food`	system asks user to confirm the value of food
`select_area`	system asks user to choose from two values of area
`select_pricerange`	system asks user to choose from two values of price range
`select_food`	system asks user to choose from two values of food
`inform_1`	system informs one slot-value with prob. > threshold
`inform_2`	system informs two slot-values with prob. > threshold
`inform_3`	system informs three slot-values with prob. > threshold
`inform_byname`	system informs the name of an entity
`inform_alternatives`	system informs an alternative entity
`inform_requested`	system informs the value of the user-requested slots
`inform_more`	system informs that there are more matched entities
`reqmore`	system requests more information from the user
`repeat`	system repeats the previous sentence
`bye`	system says good bye
`restart`	system restarts the conversation

The dimension of $\mathcal{F}_{617}$ contains two independent sources of domain dependent variability: the number of slots/names and the slot cardinality (the number of values within a slot). To eliminate the domain dependency, the issue of cardinality is managed by replacing the full distributions over each slot ('slot beliefs') with some predefined set of summary statistics of each distribution. Here *normalised entropy* η is used:

$$\eta(s) = -\sum_{v \in V_s} \frac{p(v)log(p(v))}{log|V_s|}, \tag{4.6}$$

where s is the slot considered, v is the value that can be taken within the set V_s, and the denominator $log|V_s|$ represents the maximum of the slot's entropy $H(s)$ ($max\{H(s)\} = log|V_s|$), which is also often referred to as the *information length*.

The remaining variability pertains to the differing number of slots across domains. For the system that is experimented with, there is a fixed set of N_a system actions available for each slot $\mathcal{S}$ (e.g. *'request-$\mathcal{S}$'* and *'confirm-$\mathcal{S}$'*). If a set has N_s slots, then the variability is in the fact that the set will have N_s slot entropies, plus $N_s \times N_a$ slot dependent system actions.

This problem is mitigated by imposing a limit on the number of slots a domain can have, and padding out the slot distribution summary (normalised entropy) components along with the domain dependent system actions up to this limit. The limit in the experiments is set to a maximum of 6 informable slots, making this feature have a length of 74, which is represented as $\mathcal{F}_{74}$. It is important to maintain an order to the components of this representation in order to give the RNN model the opportunity to generalise to new domains.

The number of features in $\mathcal{F}_{74}$ were 21, 38, 14, 1, respectively, for the top user act, the summarised belief state, the one-hot encoding of the domain-independent system act and scaled turn number. The latter was designed to be a light-weight feature vector applicable across multiple domains (Vandyke et al., 2015).

As described in Section 4.2, both RNN binary success classification models and RNN reward regression models were trained. To verify the effectiveness of these sequence models given the sequential inputs $\mathbf{f}_{1:N}$, a multilayer perceptron (MLP) binary success classification models with one hidden layer of size 300 was tested. This model can only handle fixed-length inputs, thus the averaged dialogue feature $\mathbf{f}_{avg} = \frac{\sum_i^N \mathbf{f}_i}{N}$ was computed as the model input. In addition, a count-based method was also proposed, which calculated the success statistics for dialogues of different turn numbers. The prediction at test time for a given dialogue is then the success ratio of the corresponding turn numbers in the training data.

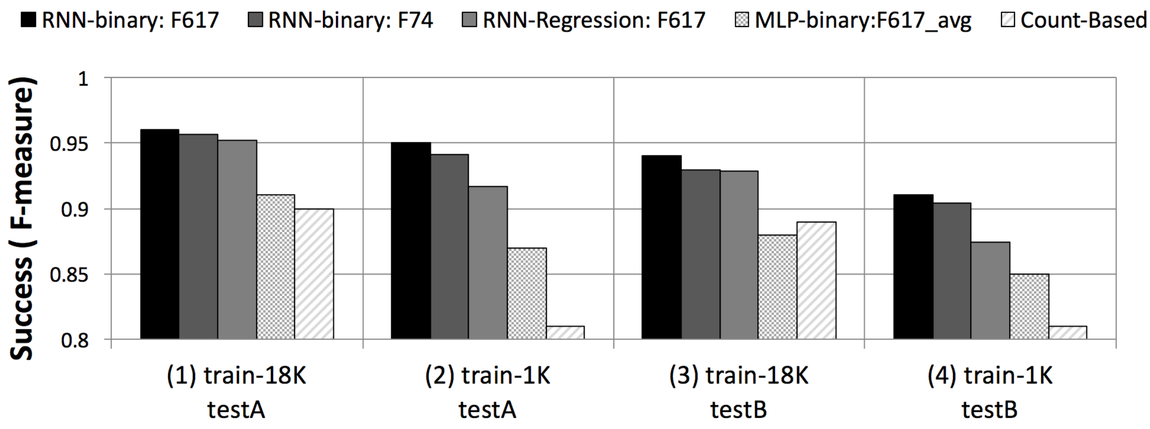


Fig. 4.4 Prediction of the proposed model trained on 18K and 1K dialogues and tested on sets *testA* and *testB* (see text). Results of success/failure label F-measure (left axis) are represented as bars.

Figure 4.4 illustrates the performance of each comparing method, where the y-axis is the F-measure of the success classification. All sub-figures show that the RNN models (three solid bars on the left of each sub-figure) outperformed all other methods. The count-based method shows that the number of turns is highly correlated to dialogue success, while the additional features further helped to improve the model accuracy as shown in other models. The comparison between the RNN binary success classification model (black bar) with features set $\mathcal{F}_{617}$ and the MLP binary success classification model (grey dotted bar) with set $\mathcal{F}_{617\_avg}$ exhibits that the fix-length averaged feature lost certain information and thus resulted in worse performance.

When using the large training set (18K, sub-figures 1 & 3), all RNN models obtained an F-score above 0.93 on success label prediction and were within $\sim \pm 0.03$ of the objective return targets on *testA* and within $\sim \pm 0.05$ on *testB*.

Without the benefit of a simulated user, it may not be possible to obtain sufficient labelled training dialogues to fully train the reward prediction model. However, the results shown in sub-figures 4.4 (2) & 4.4 (4) suggest that the proposed models were reasonably robust even with as few as 1000 training dialogues. In this case, the RNN binary classification model is the most accurate.

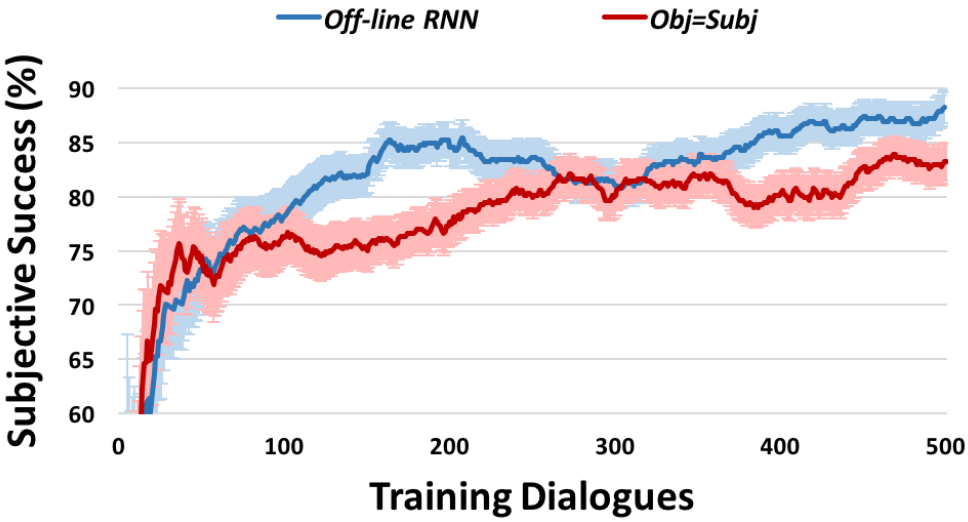


Fig. 4.5 Learning curve for the reward plotted as a function of the number of training dialogues. The baseline system (red line) updates the policy only when the *Subj* and *Obj* measures agreed. The blue line shows training using the RNN dialogue success predictor. The lightly coloured bands denote one standard error.

For the case of RNN binary success classification, both input feature sets $\mathcal{F}_{617}$ and $\mathcal{F}_{74}$ were tested. The domain specific feature set $\mathcal{F}_{617}$ only slightly outperformed the domain independent $\mathcal{F}_{74}$ set. Therefore given the desirability of domain independence, the remaining sections focus on the use of the $\mathcal{F}_{74}$ feature set.

Overall, these results suggest that an RNN model, sequentially processing turn level features, can provide useful estimates of success and/or reward. The results on set *testB* also show that the model can perform well in environments with varying error rates as would be encountered in real operating environments.

4.4.3 Policy Learning with Human Users

Based on the above results, the binary RNN classification model with the feature vector $\mathcal{F}_{74}$ was selected for training dialogue policies on-line. Two systems were trained by users recruited via the Amazon mechanical turk (AMT) service. First, an *Obj=Subj* system was trained only with the dialogues whose user *Subj* rating matched the system *Obj* measurement. This, of course, requires knowledge of each task to compute the reward and hence would not be viable for a real system. However, it serves as a useful baseline.

Second, a system was trained using only the RNN to compute the reward signal. Note that a hand-crafted system is not used for comparison here since it does not scale to larger domains and is sensitive to speech recognition errors. Three policies were trained for each system and then averaged to statistically verify the performance. Figure 4.5 shows how the success rate improves as a function of the number of available training dialogues up to a maximum of 500 dialogues. For both plots, the moving average was calculated using a window of 150 dialogues and each result was the average of the three policies to reduce noise. The figure shows that the final performance of the RNN system resulted in a comparable policy to the baseline system. However, the baseline system actually required ~ 850 dialogues (due to discarding the cases where *Obj*$\neq$*Subj*). In contrast, the RNN system was more data efficient and less costly since it used every dialogue.

The results indicate that the RNN dialogue success classifier is able to train a policy at least as well as the baseline system even though the baseline required prior knowledge of the users goal and selected only dialogues where the objective and subjective success estimates agreed.

4.5 Summary

In this chapter, an off-line reward estimator is proposed which is trained from a set of off-line simulated user data and avoids the need for prior task knowledge to compute the reward on-line. The use of the RNN model for rating success in an SDS is investigated, and the experiments have shown its effectiveness, especially the transferability of the learnt model from simulated user data to real user data.

The design of the input features to the model is based on the turn-level interaction between the user and the system. Two configurations, a domain-dependent and a domain-independent feature vector, are presented, where the latter is shown to generalise better across different domains without losing much information compared to the former.

One potential issue with this method is that the training data for the off-line reward estimator may distribute differently to the on-line user population, which may lead the policy learning towards an undesirable objective. To address this problem, Chapter 5 provides an alternative solution for practical on-line policy learning.

CHAPTER 5

Policy Learning with On-line Active Human Reward Estimator

5.1 Motivation

Several approaches, including the work in Chapter 4, have been adopted for learning a dialogue reward model given a corpus of annotated dialogues. For example, Yang et al. (2012) used collaborative filtering to infer user preferences. It has also been demonstrated that there is a strong correlation between an expert's user satisfaction rating and dialogue success (Ultes and Minker, 2015). However, all these methods assume the availability of accurate dialogue annotations such as expert ratings, which in practice are hard to obtain. Furthermore, the data for training the off-line reward predictor described in Chapter 4 may be a poor match to the user population used for on-line policy optimisation.

One effective way to mitigate the effects of annotator error in lower quality annotations is to obtain multiple ratings for the same data, and several methods have been developed to guide the annotation process using uncertainty models (Dai et al., 2013; Lin et al., 2014). Active learning is particularly useful for determining when an annotation is needed (Settles, 2010; Zhang and Chaudhuri, 2015), and it is often implemented using Bayesian optimisation (Brochu et al., 2010). For example, Daniel et al. (2014) exploited a pool-based active learning method for a robotics application.

They queried the user for feedback on the most informative sample collected so far and showed the effectiveness of this method. Active learning coupled with GP regression has been previously used for dialogue policy optimisation and shown to be more sample efficient than alternative methods (Gašić and Young, 2014).

Since humans are better at giving relative judgements than absolute scores, another related line of research has focussed on preference-based approaches to RL (Cheng et al., 2011). For example, in one study, users were asked to provide rankings between pairs of dialogues (Sugiyama et al., 2012). However, this is also costly and does not scale well in real applications.

An alternative is to learn a predictor directly on the real user population in parallel with policy optimisation, with the label annotation provided by subjective user feedback. However, as noted above, subjective user ratings, especially ones collected via crowd-sourcing, tend to be inaccurate (Gašić et al., 2011), potentially intrusive and hard to collect since users will often just 'hang-up' as soon as they have the information they need. In this chapter, a GP-based reward estimator is described which uses active learning to limit intrusive requests for feedback and a noise model to mitigate the effects of inaccurate feedback (Su et al., 2016b).

The proposed system framework is depicted in Figure 5.1. It is divided into two main parts: a dialogue embedding function, and an active reward model to obtain user feedback and predict dialogue success. When each dialogue ends, a sequence of turn-level features $\mathbf{f_t}$ is extracted and fed into an embedding function σ to obtain a fixed-dimension dialogue representation $\mathbf{d}$ that serves as the input to the reward model. This reward is modelled as a GP which for every input $\mathbf{d}$ provides an estimate of task success along with an estimate of the uncertainty. Based on this uncertainty, the reward model decides whether to query the user for feedback. It then returns a reinforcement signal to update the dialogue policy π, which is trained using the GP-SARSA algorithm, as described in Section 4.4.1.

The reward model and the dialogue policy are being jointly optimised during the sequence of dialogues. Initially, the joint learning process is supervised by the

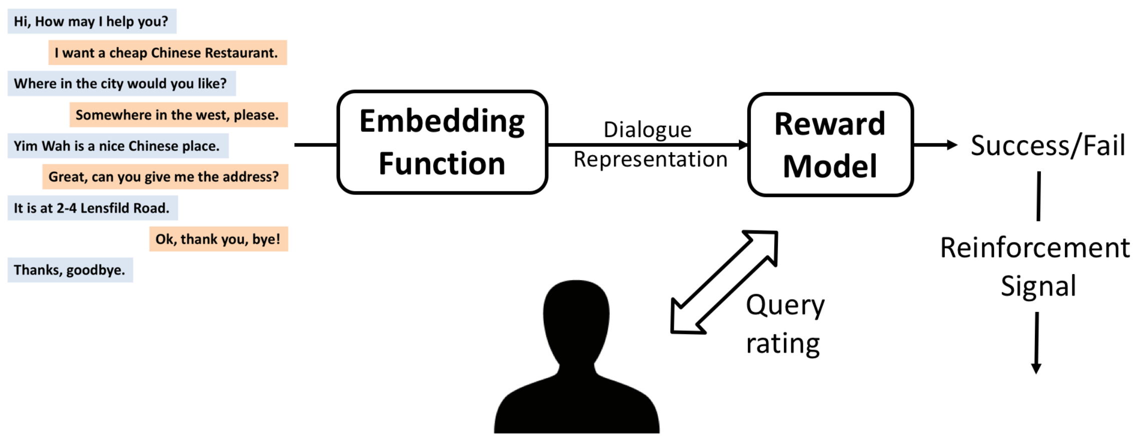


Fig. 5.1 Schematic depiction of the on-line reward estimator. The two main system components, dialogue embedding creation, and reward modelling, based on user feedback, are described in Section 5.2 and 5.3.

user feedback. Once learning is underway, the user supervision becomes lighter and ultimately the learning becomes essentially unsupervised.

5.2 Dialogue Embeddings

The use of embedding functions for sequence modelling has recently gained attention especially for word representations, and it has boosted performance on several natural language processing tasks (Levy and Goldberg, 2014; Mikolov et al., 2013; Turian et al., 2010). Embedding has also been successfully applied to machine translation (MT) where it enables varying-length phrases to be mapped to fixed-length vectors using an RNN Encoder-Decoder (Cho et al., 2014). Here an embedded representation of the sequence of dialogue turn-level features is computed to drive the reward model.

Two methods of generating dialogue embeddings have been explored, a *supervised* and an *unsupervised* approach. Once computed, these embeddings are used as the observations for the reward model described in Section 5.3.

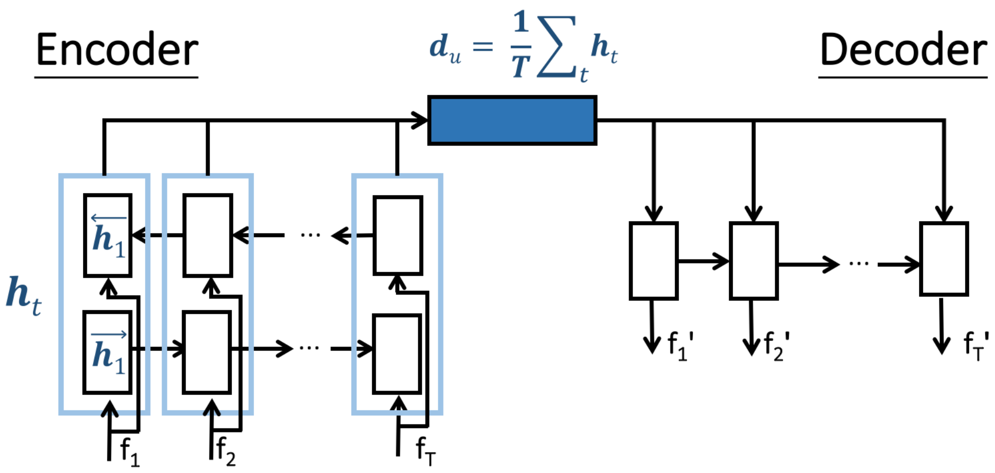


Fig. 5.2 Unsupervised dialogue embedding using the LSTM Encoder-Decoder.

5.2.1 Supervised Dialogue Embedding

In this case, the RNN in Section 4.2 is used whereby the hidden layer $\mathbf{h}_T$ at the final turn T serves as the supervised dialogue representation $\mathbf{d}_s$ to summarise the entire dialogue in a single fixed-length vector. Note, however, that as with the off-line reward estimator described in 4.2, this approach requires training data which is expensive to collect and may be biased concerning the target population.

5.2.2 Unsupervised Dialogue Embedding

To avoid the need for labelled training data, an encoder-decoder structure can be used to generate dialogue representations. The basic set-up is illustrated in Figure 5.2. Turn-level feature sequences $\mathbf{f}_t$ are input to the model, encoded and then decoded to reconstruct the input. The encoder is a bi-directional long short-term memory network (BiLSTM) (Graves et al., 2013a; Hochreiter and Schmidhuber, 1997), which considers the sequential information from both directions of the input data, computing the *forward* hidden sequences $\overrightarrow{\mathbf{h}_{1:T}}$ and the *backward* hidden sequences $\overleftarrow{\mathbf{h}_{T:1}}$ while iterating over all input features $\mathbf{f}_t$, $t = 1,...,T$:

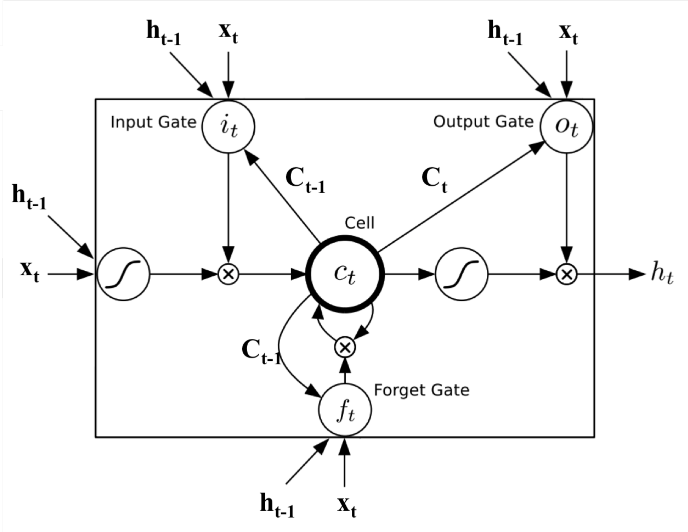


Fig. 5.3 System architecture of long short-term memory (LSTM)

$$\overrightarrow{\mathbf{h}_t} = LSTM(\mathbf{f}_t, \overrightarrow{\mathbf{h}_{t-1}})$$
$$\overleftarrow{\mathbf{h}_t} = LSTM(\mathbf{f}_t, \overleftarrow{\mathbf{h}_{t+1}}),$$

where $LSTM$ (Hochreiter and Schmidhuber, 1997) includes the memory cell c_t and element-wise multiplication gates (input i, forget f and output o). These control the information flow within the hidden layer to capture longer term dependencies than the basic RNN. Its model and equations are shown in Figure 5.3 and as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1}) \quad \text{(input gate)}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1}) \quad \text{(forget gate)}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1}) \quad \text{(output gate)}$$
$$\tilde{c_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1}) \quad \text{(candidate memory cells)}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \quad \text{(final memory cell)}$$
$$h_t = o_t \odot tanh(c_t) \quad \text{(final hidden state)}$$

where $\sigma(\cdot)$ is the non-linear activation (typically `sigmoid`), $\odot$ is element-wise multiplication, and Ws are the weight matrices between different gates i_t, f_t, o_t and the cell c_t. Note that these vectors have the same dimension as hidden layer h_t.

The dialogue representation $\mathbf{d_u}$ is then calculated as the average over all hidden sequences:

$$\mathbf{d}_u = \frac{1}{T}\sum_{t=1}^{T}\mathbf{h}_t, \tag{5.1}$$

where $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$ is the concatenation of the two directional hidden sequences.[1] The decoder is a forward LSTM that takes $\mathbf{d}_u$ as its input for each turn t to produce the sequence of features $\mathbf{f'}_{1:T}$. The encoder-decoder is trained by minimising the mean-square-error (MSE) using SGD:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}||\mathbf{f}_t - \mathbf{f}'_t||^2, \tag{5.2}$$

where N is the number of training dialogues and $||\cdot||^2$ denotes the l^2-norm.

5.3 Active Reward Learning

5.3.1 Active Learning

Querying the user for feedback is costly and may have a negative impact on the user experience. This can be reduced by using active learning based on an uncertainty estimate of the model prediction. This ensures that user feedback is only sought when the model is uncertain about its current prediction.

Active learning can be broadly categorised by its method of handling queries into either pool-based or stream-based (on-line) active learning (Settles, 2010). The pool-based method maintains a large pool of unlabelled data, and the goal is to choose the most informative sample for querying the label from the expert to better

[1] Use of a bi-directional LSTM does not cause any difficulty at run time because we are only interested in predicting success at the end of the dialogue.

estimate the model parameters. On the other hand, for the stream-based method, the unlabelled data is presented to the model one by one (on-line), and it then decides whether or not it should ask the expert to label each incoming data point.

For the case here, on-line (stream-based) active learning is utilised. The main reason is that after a user finishes the conversation with the system, it is much more natural for the system to decide whether this dialogue should be labelled by the current user rather than choosing a past dialogue (which was conducted by the system and another user) from the pool to ask the current user to label.

5.3.2 Gaussian Processes Success Classifier

A GP is a Bayesian non-parametric model that can be used for regression or classification (Bui et al., 2016; Rasmussen and Williams, 2006). As mentioned in Section 4.4.1, it is particularly appealing since it can learn from a small number of observations by exploiting the correlations defined by a *kernel function*, and it provides a measure of uncertainty of its estimates. In the context of SDSs, it has been successfully used for the RL policy optimisation (Casanueva et al., 2015; Gašić and Young, 2014) and IRL reward function regression (Kim et al., 2014).

Here we propose modelling dialogue success as a GP. This involves estimating the probability $p(y|\mathbf{d},\mathcal{D})$ that the task was successful given the current dialogue representation $\mathbf{d}$ and the pool $\mathcal{D}$ containing previously classified dialogues. We pose this as a classification problem where the rating is a binary observation $y \in \{-1,1\}$ that defines failure or success. The observations y are considered to be drawn from a Bernoulli distribution with a success probability $p(y=1|\mathbf{d},\mathcal{D})$. The probability is related to a latent function $f(\mathbf{d}|\mathcal{D}) : \mathcal{R}^{dim(\mathbf{d})} \to \mathcal{R}$ that is mapped to a unit interval by a *probit* function $p(y=1|\mathbf{d},\mathcal{D}) = \phi(f(\mathbf{d}|\mathcal{D}))$, where ϕ denotes the cumulative density function of the standard Gaussian distribution.

The latent function is given a GP prior: $f(\mathbf{d}) \sim \mathcal{GP}(m(\mathbf{d}),k(\mathbf{d},\mathbf{d}'))$, where $m(\cdot)$ represents the mean function and $k(\cdot,\cdot)$ represents the covariance function (kernel).

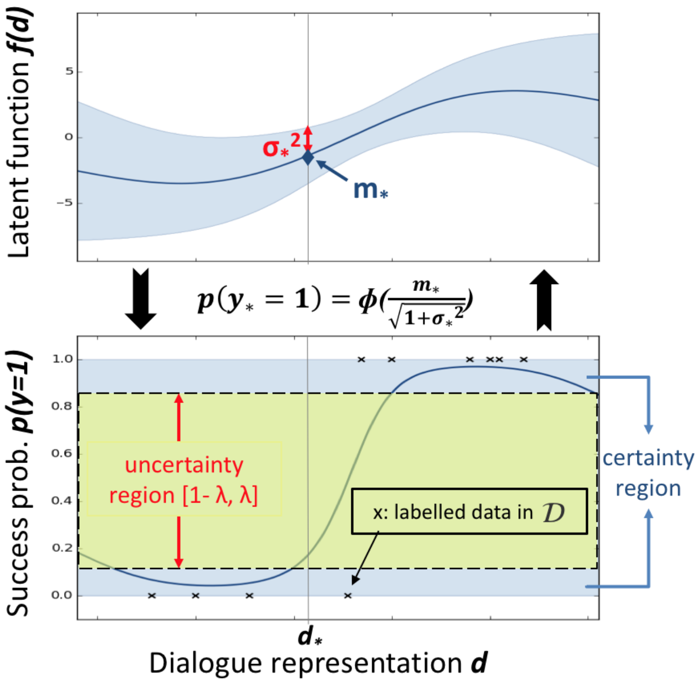


Fig. 5.4 1-dimensional example of the proposed GP active reward learning model.

Here the stationary squared exponential kernel k_{SE} is used. It is also combined with a white noise kernel k_{WN} to account for the 'noise' in users' ratings:

$$k(\mathbf{d},\mathbf{d}') = p^2 \exp(-\frac{||\mathbf{d}-\mathbf{d}'||^2}{2l^2}) + \sigma_n^2, \tag{5.3}$$

where the first term denotes k_{SE} and the second term denotes k_{WN}.

The *hyper-parameters* p, l, and σ_n can be optimised by maximising the marginal likelihood using a gradient-based method (Chen et al., 2015). Since $\phi(\cdot)$ is not linear, the resulting posterior probability $p(y=1|\mathbf{d},\mathcal{D})$ is analytically intractable. Thus, instead, approximate optimisation was performed using expectation propagation (EP) (Nickisch and Rasmussen, 2008).

A 1-dimensional example is shown in Figure 5.4. Given the labelled data $\mathcal{D}$, the predictive posterior mean μ_* and posterior variance σ_*^2 of the latent value $f(\mathbf{d}_*)$ for the current dialogue representation $\mathbf{d}_*$ can be calculated. Then a threshold interval $[1-\lambda, \lambda]$ is set on the predictive success probability $p(y_* = 1|\mathbf{d}_*, \mathcal{D}) =$

$\phi(\mu_* / \sqrt{1+\sigma_*^2})$ to decide whether this dialogue should be labelled. The decision boundary implicitly considers both the posterior mean as well as the variance.

When deploying this reward model in the proposed framework, a GP with a zero-mean prior for f is initialised and $\mathcal{D} = \{\}$. After the dialogue policy π completes each episode with the user, the generated dialogue turns are transformed into the dialogue representation $\mathbf{d} = \sigma(\mathbf{f}_{1:T})$ using the dialogue embedding function σ. Given $\mathbf{d}$, the predictive mean and variance of $f(\mathbf{d}|\mathcal{D})$ are determined, and the reward model decides whether it should seek user feedback based on the threshold λ on $\phi(f(\mathbf{d}|\mathcal{D}))$. If the model is uncertain, the user's feedback on the current episode $\mathbf{d}$ is used to update the GP model and to generate the reinforcement signal for training the policy π; otherwise, the predictive success rating from the reward model is used directly to update the policy. This process takes place after each dialogue.

5.4 Experiments: Joint Dialogue Policy and Reward Learning

5.4.1 Experimental Settings

All of the experimental work used the target domain and SDS structure described in Section 4.4.1. Policy training used the reward given by Equation 4.3, and the maximum dialogue length was set to 30. The on-line system used paid subjects recruited via the AMT service. Each user was assigned specific tasks in a given sub-domain and then asked to call the system in a similar setup to that described in (Gašić et al., 2013; Jurčíček et al., 2011). After each dialogue, the users were asked whether they judged the dialogue to be successful. Based on that binary rating, the subjective success and the average reward was calculated. An objective rating was also computed by comparing the system outputs with the assigned task specification.

5.4.2 Supervised and Unsupervised Dialogue Representations

The supervised RNN model described in Section 5.2.1 and the LSTM Encoder-Decoder model described in Section 5.2.2 were used to generate supervised and unsupervised representations $\mathbf{d}_s$ and $\mathbf{d}_u$ for each dialogue. The domain-independent feature vector $\mathcal{F}_{74}$ was used as the input of both embedding models and the target for the LSTM Encoder-Decoder model, where in the latter case, the training objective was to minimise the MSE of the reconstruction loss.

For the supervised embedding, the size of hidden layer $\mathbf{d}_s$, was set to 32. In the unsupervised case, the sizes of $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ in the encoder and the hidden layer in the decoder were all 32, resulting in $dim(\mathbf{h}_t) = dim(\mathbf{d}_u) = 64$. SGD per dialogue was used to train each model. To prevent over-fitting, *early stopping* was applied based on the held-out validation set.

The data used to train the supervised dialogue embedding were the simulated dialogues `train-1K` and the validation set, as specified in Section 4.4.2, and `testA` served as the test set. This dataset is denoted by `sim`. The embedding function was thus effectively the intermediate product created during the training of the RNN-based off-line reward model described in Section 4.4. For the unsupervised dialogue embedding, two datasets were investigated: first, the same dataset used to train the supervised embedding as mentioned above; and second, a corpus consisting of 8565, 1199 and 650 real user dialogues in the Cambridge restaurant domain was used for training, validation and testing, respectively. This dataset is denoted by `real`. This corpus was collected in previous user trials using paid subjects recruited via the AMT service.

To visualise the effect of the embeddings, all of the test dialogues were transformed using the two embedding functions and the resulting representations $\mathbf{d}_s^{\texttt{sim}}$, $\mathbf{d}_u^{\texttt{sim}}$ and $\mathbf{d}_u^{\texttt{real}}$ were plotted using t-SNE (Van der Maaten and Hinton, 2008). The results are shown in Figure 5.5, 5.6a and 5.6b. For each dialogue sample, the shape indicates whether the dialogue was successful, and the colour indicates the length

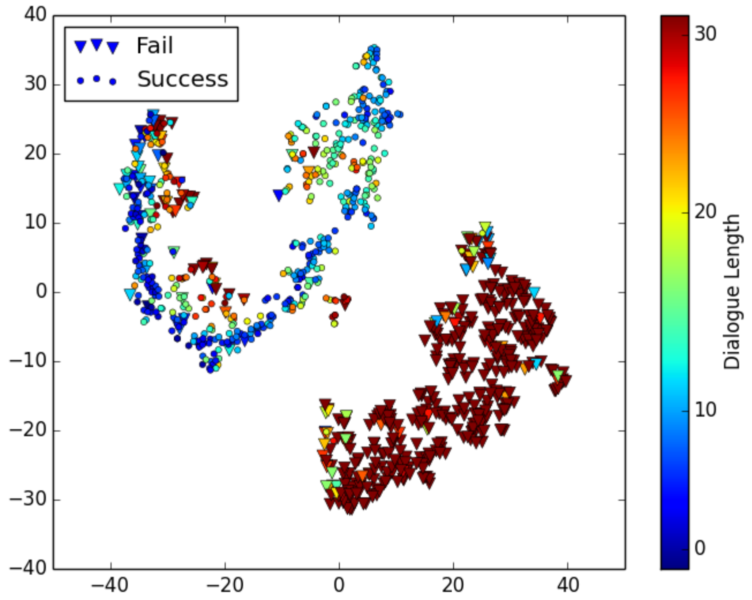
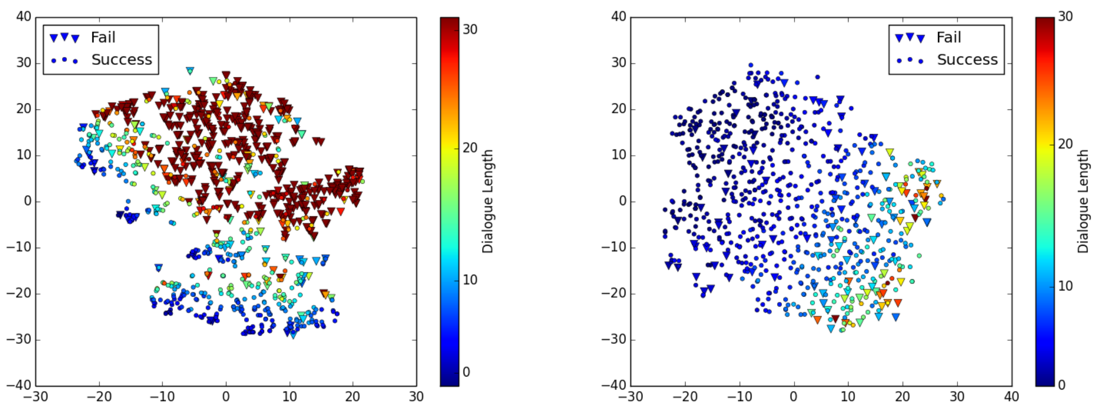


Fig. 5.5 t-SNE visualisation on the supervised dialogue representations $\mathbf{d}_s$ of the simulated data in the Cambridge restaurant domain. Labels are the subjective ratings from the users, and colours represent the total length of each dialogue.



(a) Simulated data in the Cambridge restaurant domain.

(b) Real user data in the Cambridge restaurant domain.

Fig. 5.6 t-SNE visualisation on the unsupervised dialogue representations $\mathbf{d}_u$ of different data. Labels are the subjective ratings from the users, and colours represent the total length of each dialogue.

of the dialogue (maximum 30 turns). Figure 5.5 shows that the supervised embedding function trained on simulated data separates successful and failed dialogues into two categories. It is also clear that in simulation, the successful dialogues were mostly short and the failed ones were mostly aborted when the 30-turn limit was reached.

The patterns for the unsupervised embedding trained on both data are rather different, which are shown in Figure 5.6a and 5.6b. In Figure 5.6a, the successful dialogues are distributed according to dialogue length from the bottom left (shorter dialogues) to the top right (longer dialogues). Compared to the patterns in Figure 5.5, the separation between successful and failed dialogues are less obvious since the unsupervised embedding function was trained without success labels. Similar patterns can be found in Figure 5.6b. These figures show that dialogue length was one of the most prominent features in the unsupervised dialogue representations $\mathbf{d}_u^{\mathtt{sim}}$ and $\mathbf{d}_u^{\mathtt{real}}$. Also for the dataset `real`, the failed dialogues were distributed over the length range, and the successful dialogues were on average shorter than ten turns.

Investigating these two datasets `sim` and `real`, whilst simulated and real data follow similar patterns in successful dialogues, typified by short conversations and clear information exchange between the user and the system, there is much more diversity in the real dialogues. This is especially the case for failed dialogues where real users halt the conversation as soon as they lose patience, whereas simulated users persist until the maximum turn limit is reached.

5.4.3 Dialogue Policy Learning

Given well-trained dialogue embedding functions, the proposed GP reward model operates on the embedded input space. The system was implemented using the GPy library (Hensman et al., 2012). Given the predictive success probability of each newly seen dialogue, the threshold λ for the uncertainty region was initially set to 1 to encourage label querying and annealed to 0.85 over the first 50 collected

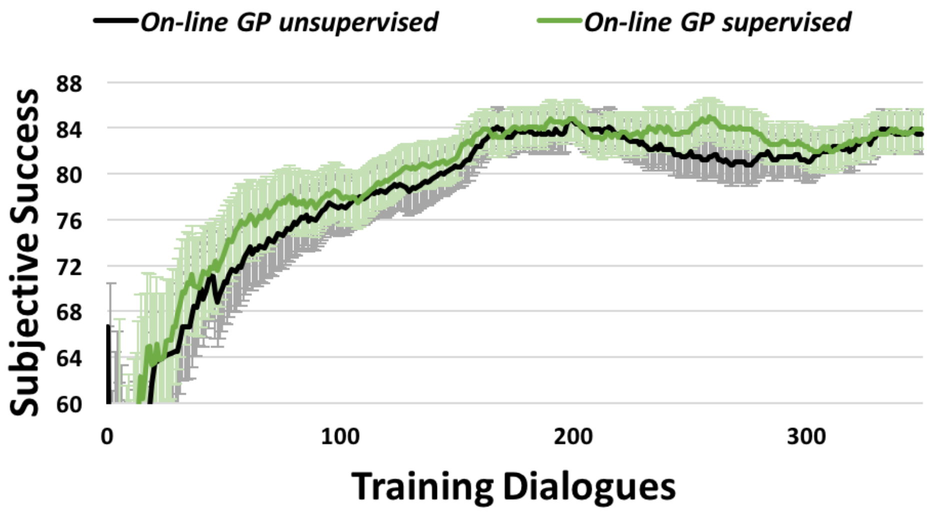


Fig. 5.7 Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation. The *on-line GP supervised*, *on-line GP unsupervised* and *Obj=Subj* systems are shown as green, black, and red lines. The light-coloured areas are one standard error intervals.

dialogues and then fixed at 0.85 thereafter. To determine a suitable threshold, various λ values were tested in simulation: 0.55, 0.65, 0.75 and 0.85. The policy with $\lambda = 0.85$ achieved the best success learning rate while requiring slightly more queries to the user compared to other λs since it was more uncertain of its prediction. Since all four models highly reduced the queries to the user with no huge difference of query numbers between each other, in real user trials, λ was set to 0.85 to ensure the best policy learning performance. Initially, as each new dialogue was added to the training set, the hyper-parameters that define the kernel structure defined in Equation 5.3 were optimised to minimise the negative log marginal likelihood using conjugate gradient ascent (Rasmussen and Williams, 2006). To prevent *overfitting*, after the first 40 dialogues, these hyper-parameters were only re-optimised after every batch of 20 dialogues.

Two systems: *on-line GP supervised* and *on-line GP unsupervised* were trained with a total of around 350 dialogues on-line by users recruited via the AMT service. Figure 5.7 shows the on-line learning curve of subjective success during training. For each system, the moving average was calculated using a window of 150 dialogues. In each case, three distinct policies were trained, and the results were averaged to reduce noise.

The figure shows that both systems had better than 80% subjective success rate after approximately 200 training dialogues. the supervised model performed slightly better during the learning process, and both converged to similar performance. This may be because the supervised embedding function was trained with knowledge of the objective success, and this is correlated to the subjective success ratings of real users. Nevertheless, after 300 dialogues, both embeddings achieve similar performance suggesting that the unsupervised embedding is perfectly adequate for the task, and it has the significant advantage that it can be trained on real data without annotations. In the following, the unsupervised method was therefore chosen for further investigation.

In addition to the above comparisons, two other systems were explored which used different methods to compute the reward:

- the *Subj* system which directly optimises the policy using only the user's subjective assessment of success whether the assessment is accurate.
- the *off-line RNN* system that uses 1K simulated data and the corresponding *Obj* labels to train an RNN success estimator as in Section 4.4.2, where the cross-entropy loss function and sigmoid activations were adopted.

For the *Subj* system rating, to focus solely on the performance of the policy rather than other aspects of the system such as the fluency of the reply sentence, users were asked to rate dialogue success by answering the following question: *Did you find all the information you were looking for?*

Figure 5.8 shows the on-line learning curve of the subjective success rating with the moving average of 150 dialogues when training the three systems: *on-line GP* (with unsupervised embedding), *Subj*, and *off-line RNN*. In each case, three distinct

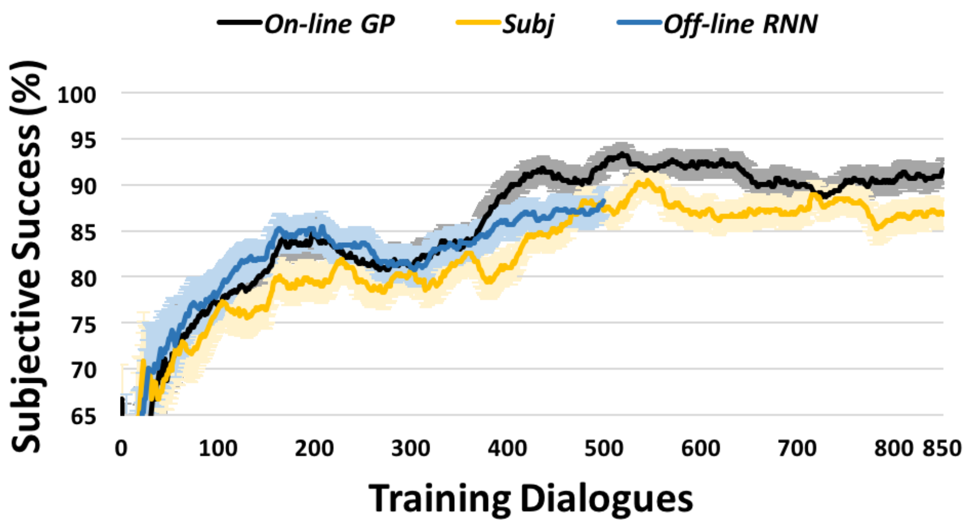


Fig. 5.8 Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation. The *on-line GP*, *Subj*, and *off-line RNN* systems are shown as black, yellow, and blue lines. The light-coloured areas are one standard error intervals.

policies were trained and the results were averaged to reduce noise. The systems were trained with 500 dialogues on-line by users recruited via the AMT service.

The figure shows that all three systems perform better than 85% subjective success rate after approximately 500 training dialogues. To investigate learning behaviour over longer spans, training for the *on-line GP* and the *Subj* systems was extended to 850 dialogues. The figure also shows that the performance in both cases is broadly flat.

Similar to the conclusions drawn in Gašić et al. (2011), the *Subj* system appears to suffer from unreliable user feedback. As with the *Obj=Subj* system in Figure 4.5, this is partly due to users forgetting the full requirements of the task and, in particular, they forget to ask for all the required information. Figure 5.8 shows that the *on-line GP* system consistently performed better than the *Subj* system, presumably, because its noise model mitigates the effect of inconsistency in user feedback. Of course, unlike crowd-sourced subjects, real users might provide more

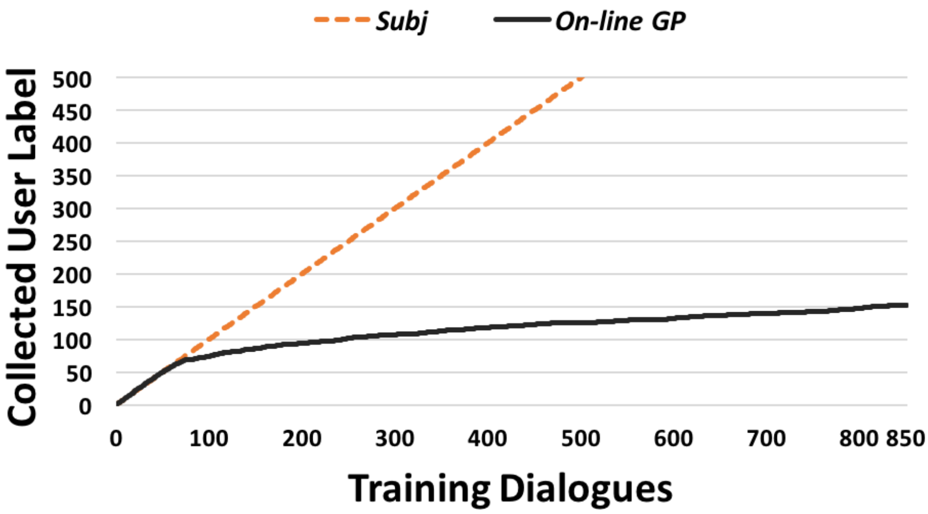


Fig. 5.9 The number of system queries for the user feedback during on-line policy learning as a function of the number of training dialogues. The orange line represents the *Subj* system, and the black line represents the *on-line GP* system.

consistent feedback, but, nevertheless, some inconsistency is inevitable and the noise model offers the needed robustness.

The advantage of the *on-line GP* system in reducing the system requests for the user feedback (i.e. the label cost) can be seen in Figure 5.9. The black curve shows the number of active learning queries triggered in the *on-line GP* system averaged across the three policies. This system required only 150 user feedback requests to train a robust reward model. On the other hand, the *Subj* system requires user feedback for every training dialogue, as shown by the dashed orange line (of course, the same applies to the *Obj=Subj* system in the previous experiment).

Certainly, the *off-line RNN* system required no user feedback at all when training the system on-line since it had the benefit of prior access to a user simulator. Its performance during training after the first 300 dialogues was, however, inferior to the *on-line GP* system.

5.4.4 Dialogue Policy Evaluation

To compare performance, the averaged results obtained between 400–500 training dialogues are shown in the first section of Table 5.1 along with one standard error. For the 400–500 interval, the *Subj*, *off-line RNN* and *on-line GP* systems achieved comparable results without statistical differences. The results of continuing training on the *Subj* and *on-line GP* systems from 500 to 850 training dialogues are also shown. The table shows that the *on-line GP* system was significantly better, presumably because it is more robust to erroneous user feedback compared to the *Subj* system.

Table 5.1 Subjective evaluation of the *off-line RNN*, *Subj* and *on-line GP* system during different stages of on-line policy learning. *Subjective*: user binary rating on dialogue success. Statistical significance was calculated using a two-tailed Student's t-test with a p-value of 0.05.

Dialogues	Reward Model	*Subjective* (%)
400-500	*off-line RNN*	89.0 ± 1.8
	Subj	90.7 ± 1.7
	on-line GP	91.7 ± 1.6
500-850	*Subj*	87.1 ± 1.0
	on-line GP	**90.9 ± 0.9***

* $p < 0.05$

5.4.5 Reward Model Evaluation

The results in Figures 5.8, 5.9 and Table 5.1 shows the advantage of the proposed reward model for policy learning. This section further investigates the accuracy of the *on-line GP* model in predicting the subjective success rate. An evaluation of the proposed model between 1 and 850 training dialogues is shown in Table 5.2.

Since three reward models were learnt, each with 850 dialogues, there were a total of 2550 training dialogues. Of these, the *on-line GP* model queried the user for feedback a total of 454 times, leaving 2096 dialogues for which learning relied on the reward model's prediction. Therefore, for a fair comparison, results shown in the table are averaged over 2096 dialogues.

The table shows that there was a significant imbalance between successful and failed dialogues since the policy was improving along with the training dialogues. This lowered the recall on failed dialogue prediction is expected since the model is biased toward successful dialogues. Overall, however, performance as judged by F-measure appears to be adequate.

Table 5.2 Statistical evaluation of the prediction of the *on-line GP* systems with respect to *Subj* rating.

Subj	Precision	Recall	F-measure	Number
Fail	1.00	0.52	0.68	204
Suc.	0.95	1.00	0.97	1892
Total	0.96	0.95	0.95	2096

5.4.6 Example Dialogues

The key benefits of the *on-line GP* reward model compared to other models are its robustness to noise and efficient use of user supervision. Since the four systems compared above differ only in the design of reward model (learning objective), their on-line behaviours were broadly similar.

Two example dialogues between users and the *on-line GP* system are listed in Table 5.3 to illustrate how the system operates under different noise conditions. The user's subjective rating and the rating determined by the *on-line GP* reward model are also shown. The labels '*n*-th ASR' and '*n*-th SEM' indicate the n-th most likely hypotheses from the speech recogniser and the semantic decoder, respectively.

5.4.7 On-line v.s. Random Queries for GP Reward Estimation

To verify the benefit of the uncertainty measurement for on-line active user label querying, the proposed model was tested in simulation in comparison to the case where the user label was queried every other dialogue. The unsupervised dialogue

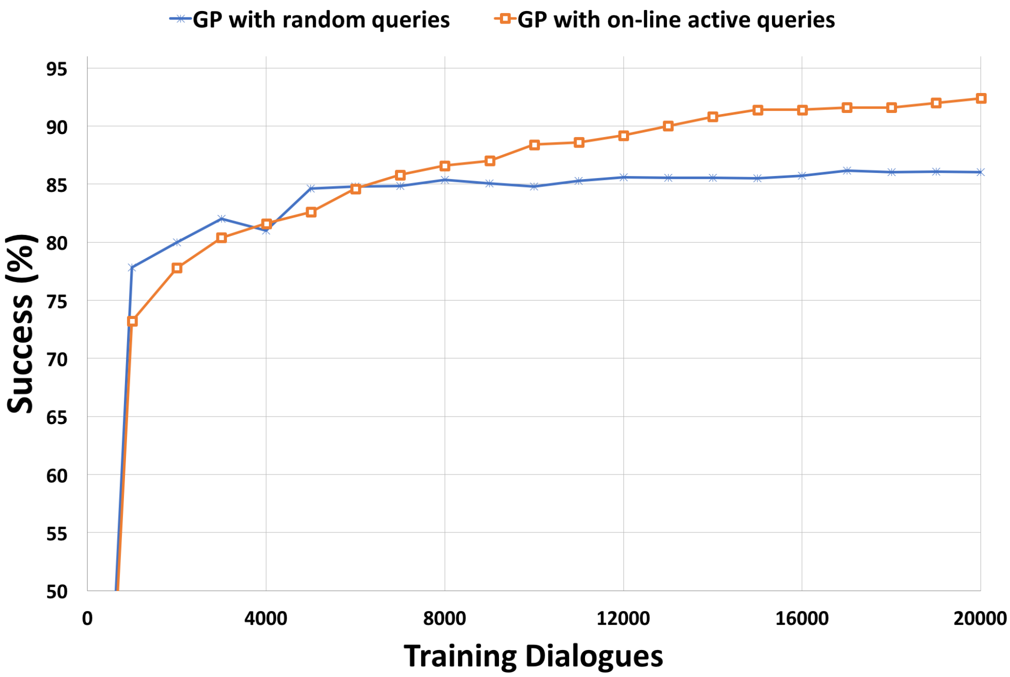


Fig. 5.10 Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation in the simulation. The *GP with on-line active queries* and *GP with random queries* systems are shown as orange and blue lines.

embedding function was utilised for both policies and the query threshold λ for the proposed model was set to 0.85. To simulate the noisy label from the user, the simulated user gave the wrong success label 35% of all time. Also, to prevent *overfitting*, after the first 40 dialogues, these hyper-parameters in Equation 5.3 were only re-optimised after every batch of 20 dialogues. Figure 5.10 illustrates the success rate learning curve of the two policy. One can clearly see that the blue line: GP with random queries (which queried the user label every other dialogue) saturated at a sub-optimal policy while the orange curve continued to grow its success rate up to around 93%. This indicates that the GP reward model's uncertainty estimation indeed helped to figure out which useful user success labels to ask for. On the other hand, the random sample from the user label inevitably confused the GP reward model with inconsistent success labels.

5.5 Summary

Computing a reliable reward function for training a dialogue policy on-line using RL is difficult since the user's objective is usually unknown. Directly querying the user at the end of each dialogue is equally problematic, since the resulting feedback is often inaccurate and repeated, querying for feedback can be intrusive. This chapter explores the design of an on-line learning framework, whereby the dialogue policy is jointly trained alongside the reward function modelled as a GP with active learning. To provide a fixed size input space for the GP, variable length dialogue turn sequences are embedded using both a supervised RNN and an unsupervised RNN-based encoder-decoder. Both perform similarly, but the latter has the advantage of requiring no task success annotation and can therefore be trained on unlabelled corpora.

This on-line reward model enables stable policy optimisation by robustly modelling the inherent noise in real user feedback and uses active learning to minimise the number of feedback requests to the user. The proposed on-line reward model achieved efficient policy learning and better performance compared to other state-of-the-art methods in the Cambridge restaurant domain. A key advantage of this Bayesian model is that its uncertainty estimate allows active learning and noise handling in a natural way. This mitigates the two key problems of requesting direct user feedback: it minimises the intrusive nature of asking for feedback, and it successfully filters inaccurate user responses.

Overall, the techniques developed in this chapter enable for the first time viable approaches to on-line learning in deployed real-world dialogue systems. Dialogue policies can be learned from scratch without requiring the collection and annotation of bootstrap training data or the construction of a user simulator.

The reward function studied here is focussed primarily on task success, and this may be too simplistic for many commercial applications. Further work may, therefore, be needed in conjunction with human interaction studies to identify and incorporate the further dimensions of dialogue quality needed to optimise user satisfaction. Nevertheless, task success is likely to remain the principal component of user satisfaction for the majority of applications; and hence the work reported here should be a major step forward in the practical deployment of systems which can learn and improve on-line without manual intervention.

On the other hand, another issue of the current RL-based dialogue policy is the sparsity of the given reward from the world (user). The system is only able to receive the most important information of the dialogue success by the end of the conversation and thus learns in-efficiently. Consequently, the next chapter focuses on enriching the reward signal to achieve more efficient learning.

Table 5.3 Example dialogues between on-line users and the proposed *on-line GP* system.

-Low noise condition- (User Subjective Evaluation: *Success*, **on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, what kind of food would you like?
Human	: [Top ASR] Yeah I want to find a restaurant that serves European food.
	[Top SEM] inform(food=european,type=restaurant)
Machine	: Hotel du Vin and Bistro is a nice place. It serves European food.
Human	: [Top ASR] Uh what is the address and phone number?
	[Top SEM] request(phone,addr)
Machine	: Hotel du Vin and Bistro is a nice place. Their address is 15-19 Trumpington Street and their phone is 01223 227330.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.
-High noise condition- (User Subjective Evaluation: *Success*, **on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, What kind of food would you like?
Human	: [Top ASR] And I want to find an expensive restaurant it should be in the south part of town.
	[Top SEM] inform(area=south,pricerange=expensive,type=restaurant)
Machine	: Gourmet burger kitchen is a nice place it is in the expensive price range.
Human	: [Top ASR] And do you have something else?
	[Top SEM] null()
	[2nd SEM] reqalts()
Machine	: Did you say you are looking for a venue in the area of south ?
Human	: [Top ASR] Yes.
	[Top SEM] affirm()
Machine	: Frankie and Benny's is a nice place, it is in the south part of town and it is in the expensive price range.
Human	: [Top ASR] Okay can I have the uh address phone number called for any.
	[Top SEM] request(addr)
	[2nd SEM] request(phone,addr)
Machine	: Frankie and Benny's is a nice place. Their address is Clifton Way and their phone number is 01223 412430.
Human	: [Top ASR] What is the pub called for free you any?
	...
	[5th ASR] What is the post code for free you any?
	[Top SEM] request(postcode)
Machine	: Frankie and Benny's is a nice place. Their postcode is C.B 1, 7 D.Y.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.

CHAPTER 6

Accelerating Policy Learning with Reward Shaping

6.1 Motivation

The mathematical explanation and literature review of **reward shaping** has been presented in Section 3.2.6; in this section, its usage is developed with the goal of speeding up the policy learning of SDS in on-line settings via the RNN models. An alternative form of RNN setting compared to the one in Figure 4.2 of Section 4.2 is utilised with the introduction of new training constraints to combine the three following features:

1. Diffuse the dialogue level ratings down to the turn level.
2. Add reward shaping information for faster policy learning.
3. Avoid using prior task knowledge, which is in practice unavailable on-line.

The core idea here is using the RNN models to determine adequate values (information) to add as a potential ϕ in Equation 3.29 to the existing learning systems' reward function in order to guide the system to achieve the desired objective faster.

6.1.1 Operating environment and dialogue features

In all experiments, the Cambridge restaurant domain was used where the domain ontology and shared components within the SDS are illustrated in Section 4.4.1. GP-SARSA (Gašić and Young, 2014) was employed for dialogue policy optimisation, and the summary action space contains 20 actions. A per-turn reward of -1 was used and the final reward of 20 was assigned if the dialogue was successful; otherwise, a final reward of 0 was assigned.

As mentioned in Sections 4.3 and 4.4.2, the feature vector of full size $\mathcal{F}$ within the ontology has been shown to be redundant, whereas the domain independent features $\mathcal{F}_{74}$ enable cross-domain applications whilst only slightly sacrificing prediction accuracy. Here, since only a single domain is considered, the feature vector applied is based on $\mathcal{F}_{74}$ but has more domain dependent elements. For instance, rather than using the single slot-independent action (e.g. '*request*'), it is mapped back to the slot dependent system actions (e.g. {'*request-*$\mathcal{S}_1$', ..., '*request-*$\mathcal{S}_{N_s}$'}), providing the feature vector with the exact slot the system is focussing on. In addition, the normalised entropy representing the slot statistics, as in Equation 4.6, was also replaced by its original distribution over each value within a slot. The resulting feature vector is in the form as follows: one-hot encodings of each of the system action and most likely user action (as estimated by the belief tracker), full belief state (distributions over slot values) and normalised turn number (by 30). It is of size 147, and is used in all of the experiments reported in this chapter.

6.1.2 RNN model structures for reward shaping

RNN models with basic/LSTM/GRU recurrent units are adopted to manage the variable length of each dialogue by updating a hidden layer $\mathbf{h}_t$ with the input feature $\mathbf{f}_t$ at each turn t. The architecture of the RNN with the basic recurrent unit and the LSTM recurrent unit are described in Section 4.2 and Section 5.2.1.

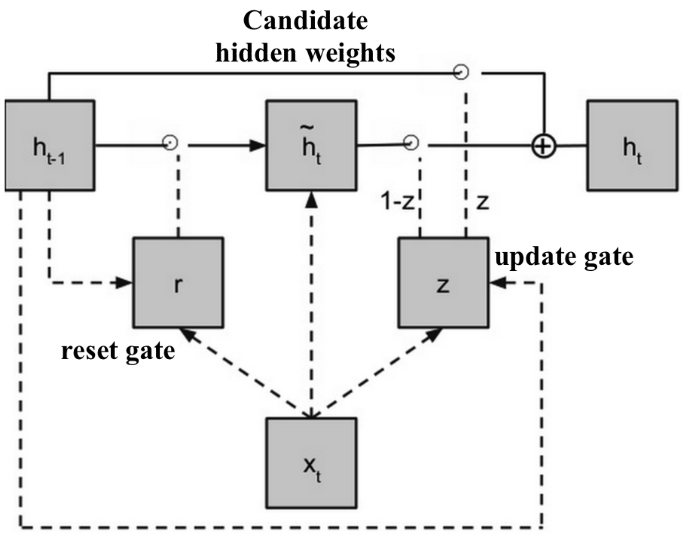


Fig. 6.1 System architecture of gated recurrent unit (GRU).

Looking into the operating mechanisms within the GRU recurrent units (Chung et al., 2014), instead of maintaining a memory cell in LSTM as described in Section 5.2.1, it directly utilises a reset gate r_t and an update gate z_t to control the memory flow between hidden layers in different time-stamps. This is illustrated in Figure 6.1 and detailed as follows:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1}) \qquad \text{(update gate)}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1}) \qquad \text{(reset gate)}$$

$$\tilde{h_t} = tanh(W_{xh}x_t + r_t \odot W_{hh}h_{t-1}) \qquad \text{(candidate hidden state)}$$

$$h_t = (1 - z) \odot \tilde{h_t} + z \odot h_{t-1} \qquad \text{(final hidden state)}$$

There has not been a clear conclusion drawn on the comparison between the LSTM and GRU to date (Jozefowicz et al., 2015). Despite the extra complexity introduced, these two variations have been found to improve the performance of the basic RNN in many NLP tasks (Kumar et al., 2015; Sundermeyer et al., 2012).

The role of the RNN model is to solve the regression problem of predicting the scalar return of each completed dialogue. At every turn t, an input feature

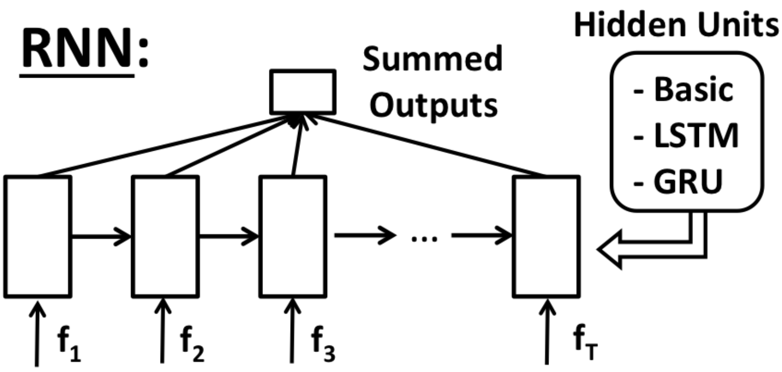


Fig. 6.2 RNN with three types of hidden units: basic, LSTM and GRU. The feature vectors $\mathbf{f}_t$ extracted at turns $t = 1, \ldots, T$ are labelled $\mathbf{f}_t$.

f_t is extracted from the belief/action pair and used to update the hidden layer h_t. From dialogues generated by a simulated user (Schatzmann and Young, 2009), supervised training pairs are created that consist of the turn level sequence of these feature vectors f_t along with the scalar dialogue return as scored by an objective measure of task completion. While the RNN models are trained on dialogue level supervised targets, it is hypothesised that their subsequent turn level predictions can guide policy exploration by acting as informative reward shaping potentials ϕ.

To encourage good turn level predictions, all three RNN variants are trained to predict the dialogue return not just with the final output of the network but with the constraint that their scalar outputs from each turn t should sum to predict the return for the whole dialogue. The loss function and the connection between the output layer and the hidden layers are different from that in Section 4.2; because in that previous case, the goal is to predict as precisely as possible the final dialogue-level rating, while the focus in this case is to determine accurate turn-level ratings. This RNN model structure is shown in Figure 6.2, and a mean-square-error (MSE) loss on a per-dialogue basis is used:

$$\mathrm{MSE} = \left(\sum_{t=0}^{T-1} r_t - \mathcal{R} \right)^2, \tag{6.1}$$

where the current dialogue has T turns, $\mathcal{R}$ is the dialogue return and training target, and r_i is the scalar prediction output by the RNN model at each turn. The trained RNN models are then used directly as the reward shaping potential function ϕ, using the RNN scalar output given the input s_t at each turn t. As shown in Equation 3.29, the turn-level shaping reward is thus: $F(b_t, a_t, b_{t+1}) = \gamma\phi(b_{t+1}) - \phi(b_t)$. As a consequence, the final composite reward $R'(b_t, a_t) = R(b_t, a_t) + F(b_t, a_t, b_{t+1})$ at each turn t is used for the policy learning, where $R(b_t, a_t)$ is the original environmental reward.

6.2 Experiments

6.2.1 Neural Network Training for Return Prediction

This section presents the results of training the 3 RNN models on the simulated user dialogues. In all cases the hidden layer contained 100 units with a sigmoid non-linearity and used SGD (per dialogue) for training. The training, validation and test set used here are similar to those in Section 4.4.2: `train-18K`, `train-1K`, `testA`, `testB`, and again the separate validation sets of 1K dialogues were used for controlling overfitting while training.

The Root-MSE (RMSE) result of predicting the dialogue return are depicted in Figure 6.3. Note that the predictions were unbounded and the averaged return is around 8–10. Hence the worst RMSE could be as more than 100. In all cases, the RMSEs are around 5–7, which indicates high similarity of the patterns between the sum of predicted turn-level ratings to the true return values. The results are encouraging because one of the crucial characteristics of reward shaping is assigning an appropriate 'extra' turn-level penalty or reward to the learning system for speed-up. The models with LSTM and GRU units achieved a slight improvement in most cases over the basic RNN. Notice that the model with GRU even reached comparable results when trained with 1K training data compared to 18K. The results from the 1K training set indicate that the model can be developed from

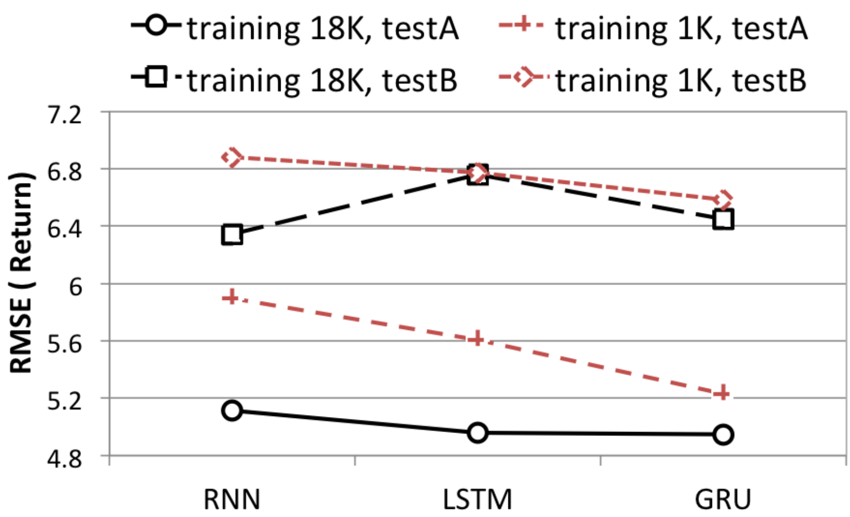


Fig. 6.3 RMSE of return prediction by using RNN/LSTM/GRU, trained on 18K and 1K dialogues and tested on sets *testA* and *testB* (see text).

limited data. The results on set *testB* also show that the models can perform well in situations with varying error rates as would be encountered in real operating environments. The dataset could also be created from human annotations, which would avoid the need for a simulated user. In the following sections, the focus is to examine the RNN-based reward shaping for the policy training with a simulated user and human users.

6.2.2 Policy Learning with a Simulated User

Since the aim of reward shaping is to enhance the speed of policy learning, the emphasis is placed on the first 1000 training dialogues. Figure 6.3 shows that the RNN with GRU recurrent unit attained slightly better performance than the other two RNN models, albeit with no statistical significance. Thus for the clearer presentation of the policy training results, only the GRU results are plotted, using the model trained on 18K dialogues.

To show the effectiveness of using RNN with GRU for predicting reward shaping potentials, the proposed method is compared with the hand-crafted (HDC)

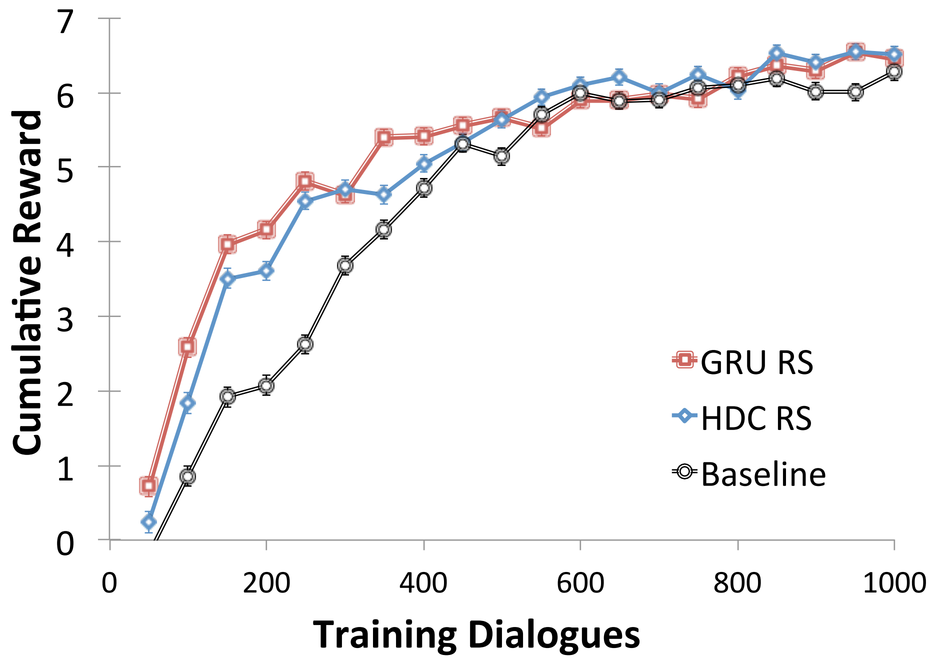


Fig. 6.4 Policy training with a simulated user with (GRU/HDC) and without (Baseline) reward shaping (RS). Standard errors are also shown.

method for reward shaping used in (Ferreira and Lefèvre, 2013) that requires prior knowledge of the user's task and a baseline policy using only the environmental reward. Figure 6.4 shows the learning curve of the cumulative reward (return) for the three systems. After every 50 training iterations, each system was tested with 1000 dialogues and averaged over ten policies. The simulated user's SER was set to 15%.

We see that reward shaping indeed provides the agent with more information and increases the learning speed within training dialogues 0–500, while achieving similar performance after 500 training dialogues. Moreover, the proposed RNN method outperforms the hand-crafted system in the early stages of the training, while also being able to be applied on-line.

To inspect how exactly the proposed model helps in each turn, Figure 6.5 provides a qualitative analysis of the turn-level prediction of the proposed GRU model on one successful simulated dialogue example, where the system provided the

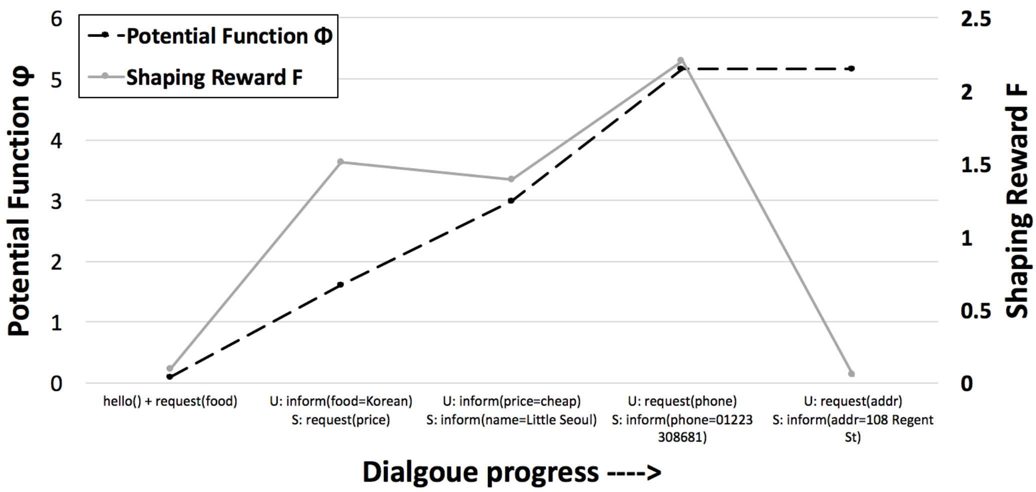


Fig. 6.5 Turn-level predictions of the GRU model in a successful dialogue example. They serve as the potential function ϕ (black dashed line) in Equation 3.28 and the corresponding shaping rewards F (grey solid line) are also presented.

desired information the user was asking for. The black dashed line is the prediction from the GRU, which also serves as the potential function $\phi(b_t)$ mentioned in Equation 3.28. The grey solid line represents the additional shaping reward $F(b_t, a, b_{t+1}) = \gamma\phi(b_{t+1}) - \phi(b_t)$ to be added to the original reward setting. In this successful five-turn dialogue example, a matched restaurant was determined and informed given the user's search constraints. The system then provided the requested information such as phone number and address to the user. Along with this successful dialogue progress, the GRU predictions ϕ gradually increased. This illustrates that the proposed model has learned to make appropriate estimations on intermediate dialogue turns. It is worth mentioning that the GRU model has learned to predict higher $\phi(b_t)$ and led to higher shaping reward F when the system started to provide the contact information of the restaurant asked by the user. In this case, the user implicitly agreed that the system had provided his/her desired restaurant and this usually lead to a successful dialogue.

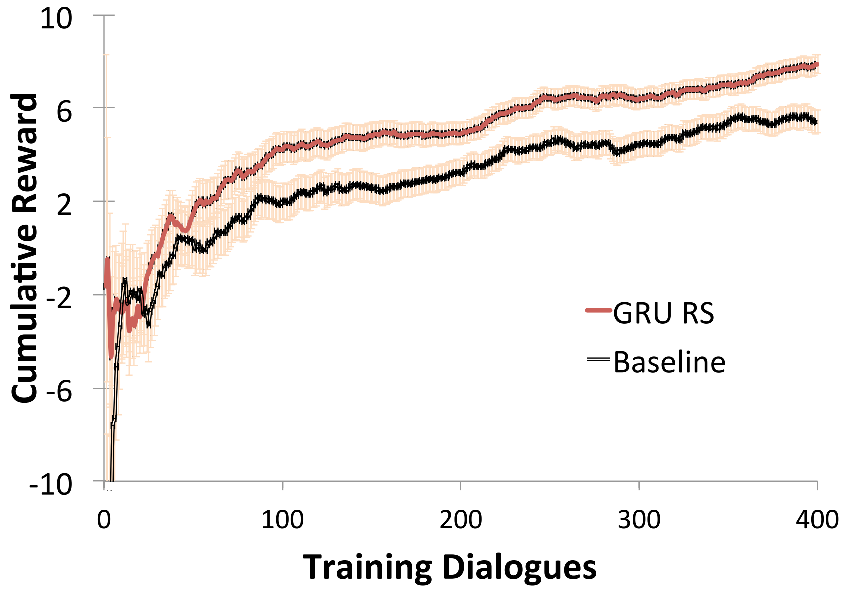


Fig. 6.6 Policy training in on-line human user scenarios for the baseline (black) and proposed GRU RS (red) systems. Standard errors are also shown.

6.2.3 Policy Learning with Real Users

Based on the above results, the same GRU model was selected for training a policy on-line with humans. Two systems were trained with users recruited via AMT: a baseline was trained with only the environmental reward, and another system was trained with an additional shaping reward predicted by the proposed GRU. Learning began with a random policy in all cases.

Figure 6.6 shows the on-line learning curve of the reward when training the systems with 400 dialogues. The moving average was calculated using a window of 100 dialogues and each result was averaged over three policies to reduce noise. Consistent with the results based on user simulation in Section 6.2.2, by adding the RNN based shaping reward, the policy learnt quicker in the important initial phase of policy learning.

6.3 Summary

This chapter has shown that RNN models can be trained to predict the dialogue return with the constraint that subsequent turn level predictions act as good reward shaping signals that are effective for accelerating policy learning on-line with real users. As in many other applications, we found that gated RNNs such as LSTM and GRU perform slightly better than basic RNNs.

In the work described in this section, the RNNs were trained using a simulated user, and this simulator could have been used to bootstrap a policy for use with real users. However, the supposition is that RNNs could be trained for reward prediction which are substantially domain independent and hence have wider application via domain adaptation and extension.

In addition to utilising neural network models for predicting reward, in recent years deep neural networks have been applied to RL-based policy optimisation (deep RL) and have shown success in various large-scale tasks (Mnih et al., 2013; Silver et al., 2016). However, deep RL suffers from low sample-efficiency and instability in learning. In the next chapter, several deep RL methods for learning a dialogue policy as well as proposing approaches for speeding up and improving learning stability.

CHAPTER 7

Learning Sample-efficient Deep RL Policies

7.1 Motivation

RL-based approaches to dialogue management have been actively studied and shown to be effective for many years (Lemon et al., 2006; Levin et al., 1998). As described in Section 4.4.1, data-efficient methods such as GPs have enabled systems to be trained from scratch via on-line interaction with real users (Gašić et al., 2011; Gašić and Young, 2014). GP provides an estimate of the uncertainty in the underlying function and a built-in noise model. This helps to achieve highly sample-efficient exploration and robustness to recognition/understanding errors.

However, since the computation in GP scales with the number of points memorised, sparse approximation methods such as the *kernel span* algorithm (Engel, 2005) must be used, and this limits the ability to scale to very large training sets. It is therefore questionable as to whether GP can scale to support commercial wide-domain SDS. Nevertheless, GPs provide a strong benchmark, and hence it is included in the evaluation in this chapter.

On the other hand, deep RL methods have been shown to have significant potential for dialogue policy optimisation due to their high scalability and flexibility (Gu et al., 2016; Levine et al., 2016; Mnih et al., 2013; Silver et al., 2016). However, they typically learn slowly using gradient-based methods, which is especially problematic for on-line learning with real users. To address this problem, this chapter makes two contributions:

1. improve the sample-efficiency of actor-critic RL: TRACER and eNACER.

2. efficient utilisation of demonstration data for improved early stage policy learning.

The first part focusses primarily on increasing the RL learning speed. For TRACER, trust regions are introduced to the standard actor-critic to control the step size and thereby avoid catastrophic model changes. For eNACER, the natural gradient identifies the steepest ascent direction in policy space to ensure fast convergence. Both models exploit off-policy learning with experience replay (ER) to improve sample-efficiency. These are compared with various state-of-the-art RL methods.

The second part aims to mitigate the cold start issue by using *demonstration data* to pre-train an RL model. This resembles the training procedure adopted in recent game playing applications (Hester et al., 2017; Silver et al., 2016). A key feature of this framework is that a single model is trained using both SL and RL with different training objectives but without modifying the architecture.

By combining the above, we demonstrate a practical approach to learning deep RL-based dialogue policies for new domains which can achieve competitive performance without significant detrimental impact on users in the early stage of learning.

Combining SL with RL for dialogue modelling is not new. Henderson et al. (2008) proposed a hybrid SL/RL model that, to ensure tractability in policy optimisation, performed exploration only on the states in a dialogue corpus. The policy was then defined manually on parts of the space which were not found in the corpus. A method of initialising RL models using logistic regression was also described (Rieser and Lemon, 2006). For GP-SARSA in dialogue, rather than using a linear kernel that imposes heuristic data pair correlation, a pre-optimised

Gaussian kernel learned using SL from a dialogue corpus has been proposed (Chen et al., 2015). The resulting kernel was more representative on the data distribution and achieved better performance; however, the SL corpus did not help initialise a better policy. The better initialisation of GP-SARSA has been studied in the context of domain adaptation by specifying a GP prior or re-using an existing model which is then pre-trained for the new domain (Gašić et al., 2013a).

A number of authors have proposed training a standard neural-network policy in two stages (Das et al., 2017; Fatemi et al., 2016; Su et al., 2016a; Williams et al., 2017; Zhao and Eskenazi, 2016). Asadi and Williams (2016) also explored off-policy RL methods for dialogue policy learning. All these studies were conducted in simulation, using error-free text-based input. A similar approach was also used in a conversational model (Li et al., 2016). In contrast, our work introduces two new sample-efficient actor-critic methods, combines both two-stage policy learning and off-policy RL, and tests at differing noise levels.

7.2 Neural Dialogue Management

The proposed framework addresses the dialogue management component, especially the dialogue policy, in a modular SDS. The input to the model is the belief state $\mathbf{b}$ that encodes a distribution over the possible user intents along with the dialogue history. The model's role is to select the system action a at every turn that will lead to a successful dialogue outcome. The system action is mapped into a system reply at the semantic level, and this is subsequently passed to the natural language generator for output to the user.

The semantic reply consists of three parts: the *intent* of the response, (e.g. inform), which *slots* to talk about (e.g. area), and a *value* for each slot (e.g. east). To ensure tractability, the policy selects a from a restricted action set which identifies the *intent* and sometimes a *slot*, and any remaining information required to complete the reply is extracted using heuristics from the tracked belief state.

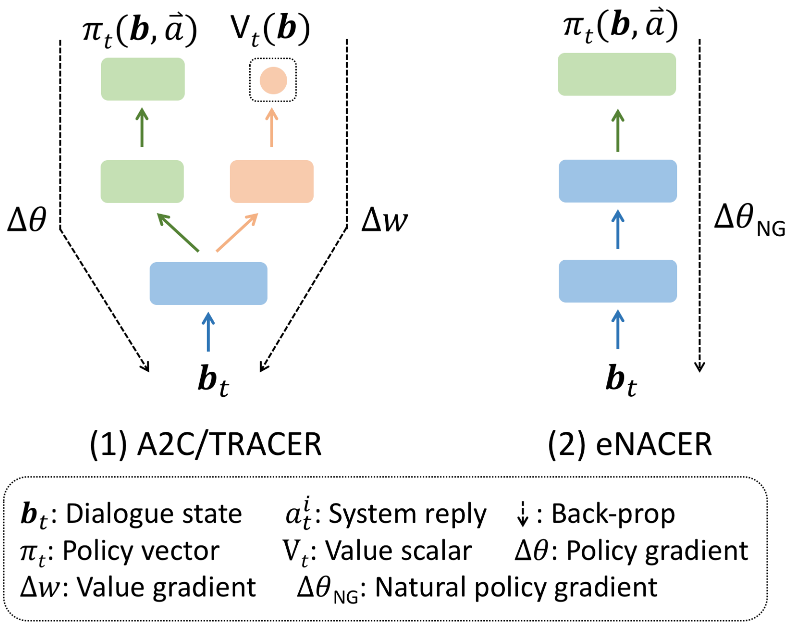


Fig. 7.1 A2C, TRACER, and eNACER architectures using feed-forward neural networks.

7.2.1 Training with Reinforcement Learning

Dialogue policy optimisation can be seen as the task of learning to select the sequence of responses (actions) at each turn which maximises the long-term objective defined by the reward function. Described in Section 3.1.5, this can be solved by applying either value-based or policy-based methods. The main difference between the two categories is that policy-based methods have stronger convergence characteristics than value-based methods. The latter often diverge when using function approximation since they optimise in value space and a slight change in value estimate can lead to a large change in policy space (Sutton et al., 2000).

Policy-based methods suffer from low sample-efficiency, high variance and often converge to local optima since they typically learn via Monte Carlo estimation (Schulman et al., 2016; Williams, 1992). However, they are preferred due to their superior convergence properties. Hence in this chapter, we focus on policy-based methods but also include a value-based method as a baseline.

Advantage Actor-Critic (A2C)

In a policy-based method, the training objective is to find a parametrised policy $\pi_\theta(a|\mathbf{b})$ that maximises the expected reward $J(\theta)$ over all possible dialogue trajectories given a starting state.

Following the *Policy Gradient Theorem* (Sutton et al., 2000), the gradient of the parameters given the objective function has the form:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|\mathbf{b}) Q^{\pi_\theta}(\mathbf{b},a)\right]. \tag{7.1}$$

Since this form of gradient has a potentially high variance, a baseline function is typically introduced to reduce the variance while not changing the estimated gradient (Sutton and Barto, 1999; Williams, 1992). A natural candidate for this baseline is the value function $V(\mathbf{b})$. Equation 7.1 then becomes the following:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|\mathbf{b}) A_w(\mathbf{b},a)\right], \tag{7.2}$$

where $A_w(\mathbf{b},a) = Q(\mathbf{b},a) - V(\mathbf{b})$ is the *advantage function*. This can be viewed as a special case of the *actor-critic*, where π_θ is the actor and $A_w(\mathbf{b},a)$ is the critic, which is defined by two parameter sets θ and w. To reduce the number of required parameters, TD errors $\delta_w = r_t + \gamma V_w(\mathbf{b}_{t+1}) - V_w(\mathbf{b}_t)$ can be used to approximate the advantage function (Schulman et al., 2016). The left part in Figure 7.1 shows the architecture and parameters of the resulting A2C policy.

The TRACER Algorithm

To boost the performance of A2C policy learning, two methods are introduced:

I. Experience replay with off-policy learning for speed-up

On-policy RL methods update the model with the samples collected via the current policy. Sample-efficiency can be improved by utilising ER (Lin, 1992) where mini-batches of dialogue experiences are randomly sampled from a "replay pool"

$\mathcal{P}$, which is a buffer with a pre-specified size maintaining the previous dialogue experiences, to train the model. This increases learning efficiency by re-using past samples in multiple updates while ensuring stability by reducing the data correlation. Since these past experiences were collected from different policies compared to the current policy, the use of ER leads to *off-policy* updates.

When training models with RL, ϵ-greedy action selection is often used to trade-off between exploration and exploitation, whereby a random action is chosen with probability ϵ otherwise the top-ranking action is selected. A policy used to generate training dialogues (episodes) is referred to as a *behaviour policy* μ, in contrast to the policy to be optimised which is called the *target policy* π.

The basic A2C training algorithm described in Section 7.2.1 is *on-policy* since it is assumed that actions are drawn from the same policy as the target to be optimised ($\mu = \pi$). In *off-policy* learning, since the current policy π is updated with the samples generated from old behaviour policies μ, an importance sampling (IS) ratio is used to rescale each sampled reward to correct for the sampling bias at each time-step t (Meuleau et al., 2000):

$$\rho_t = \pi(a_t|\mathbf{b}_t)/\mu(a_t|\mathbf{b}_t). \tag{7.3}$$

For A2C, the off-policy gradient for the parametrised value function V_w thus has the following form:

$$\Delta w^{\text{off}} = \sum_{t=0}^{T-1} \big(\bar{R}_t - \hat{V}_w(\mathbf{b}_t)\big) \nabla_w \hat{V}_w(\mathbf{b}_t) \prod_{i=0}^{t} \rho_i, \tag{7.4}$$

where $\bar{R}_t$ is the off-policy Monte-Carlo return (Precup et al., 2001):

$$\bar{R}_t = r_t + \gamma r_{t+1} \prod_{i=1}^{1} \rho_{t+i} + \cdots + \gamma^{T-t-1} r_{T-1} \prod_{i=1}^{T-1} \rho_{t+i}. \tag{7.5}$$

Likewise, the updated gradient for policy π_θ is:

$$\Delta\theta^{\text{off}} = \sum_{t=0}^{T-1} \rho_t \nabla_\theta \log \pi_\theta(a_t|\mathbf{b}_t)\hat{\delta}_w, \tag{7.6}$$

where $\hat{\delta}_w = r_t + \gamma \hat{V}_w(\mathbf{b}+_{t+1}) - \hat{V}_w(\mathbf{b}_t)$ is the TD error using the estimated value of $\hat{V}_w$.

Also, as the gradient correlates strongly with the sampled reward, reward r_t and total return R are normalised to lie in [-1,1] to stabilise training.

II. Trust region constraint for stabilisation

To ensure stability in RL, each per-step policy change is often limited by setting a small learning rate. However, setting the rate low enough to avoid occasional large destabilising updates is not conducive to fast learning.

Here, we adopt a modified Trust Region Policy Optimisation method introduced by Wang et al. (2017). In addition to maximising the cumulative reward $J(\theta)$, the optimisation is also subject to a Kullback-Leibler (KL) divergence limit between the updated policy θ and an *average policy* θ_a to ensure safety. This average policy represents a running average of past policies and constrains the updated policy to not deviate far from the average $\theta_a \leftarrow \alpha\theta_a + (1-\alpha)\theta$ with a weight α.

Thus, given the off-policy policy gradient $\Delta\theta^{\text{off}}$ in Equation 7.6, the modified policy gradient with trust region g is calculated as follows:

$$\begin{aligned} \underset{g}{\text{minimise}} \quad & \frac{1}{2}\|\Delta\theta^{\text{off}} - g\|_2^2, \\ \text{subject to} \quad & \nabla_\theta D_{KL}\left[\pi_{\theta_a}(\mathbf{b}_t)\|\pi_\theta(\mathbf{b}_t)\right]^T g \le \xi, \end{aligned} \tag{7.7}$$

where π is the policy parametrised by θ or θ_a, and ξ controls the magnitude of the KL constraint. Since the constraint is linear, a closed form solution to this quadratic programming problem can be derived using the KKT conditions. Setting

$k = \nabla_\theta D_{KL}\left[\pi_{\theta_a}(\mathbf{b}_t) \| \pi_\theta(\mathbf{b}_t)\right]$, we obtain the following:

$$g_{tr}^* = \Delta\theta^{\text{off}} - \max\left\{\frac{k^T \Delta\theta^{\text{off}} - \xi}{\|k\|_2^2}, 0\right\} k. \tag{7.8}$$

When this constraint is satisfied, there is no change to the gradient with respect to θ. Otherwise, the update is scaled down along the direction of k and the policy change rate is lowered. This direction is also shown to be closely related to the *natural gradient* (Amari, 1998; Schulman et al., 2015), which is presented in the next section.

The above enhancements speed up and stabilise A2C. We call it the trust region actor-critic with experience replay (TRACER) algorithm.

The eNACER Algorithm

Vanilla gradient descent algorithms are not guaranteed to update the model parameters in the steepest direction due to re-parametrisation (Amari, 1998; Martens, 2014). A widely used solution to this problem is to use a *compatible function approximation* for the advantage function in Equation 7.2: $\nabla_w A_w(b, a) = \nabla_\theta \log \pi_\theta(a|b)$, where the update of w is then in the same update direction as θ (Sutton et al., 2000). Equation 7.2 can then be rewritten as follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|\mathbf{b}) \nabla_\theta \log \pi_\theta(a|\mathbf{b})^T w\right] \\ &= F(\theta) \cdot w, \end{aligned} \tag{7.9}$$

where $F(\theta)$ is the Fisher information matrix. This implies $\Delta\theta_{NG} = w = F(\theta)^{-1} \nabla_\theta J(\theta)$ and it is called the *natural gradient*. The Fisher Matrix can be viewed as a correction term which makes the natural gradient independent of the parametrisation of the policy and corresponds to steepest ascent towards the objective (Martens, 2014). Empirically, the natural gradient has been found to significantly speed up convergence.

Based on these ideas, the natural actor-critic (NAC) algorithm was developed by Peters and Schaal (2006). In its episodic version (eNAC), the Fisher matrix does not need to be explicitly computed. Instead, the gradient is estimated by a least squares method given the n-th episode consisting of a set of transition tuples $\{(\mathbf{b}_t^n, a_t^n, r_t^n)\}_{t=0}^{T_n-1}$:

$$R^n = \left[\sum_{t=0}^{T_n-1} \nabla_\theta \log \pi_\theta(a_t|\mathbf{b}_t;\theta)^T\right] \cdot \Delta\theta_{NG} + C, \tag{7.10}$$

which can be solved analytically. C is a constant which is an estimate of the baseline $V(\mathbf{b})$.

As in TRACER, eNAC can be enhanced by using ER and off-policy learning, thus called eNACER, whereby R^n in Equation 7.10 is replaced by the off-policy Monte-Carlo return $\bar{R}_0^n$ at time-step $t = 0$, as in Equation 7.5. For very large models, the inversion of the Fisher matrix can become prohibitively expensive to compute. Instead, a truncated variant can be used to calculate the natural gradient (Schulman et al., 2015).

eNACER is structured as a feed-forward network with the output π as in the right of Figure 7.1, updated with natural gradient $\Delta\theta_{NG}$. Note that by using the compatible function approximation, the value function does not need to be explicitly calculated. This makes eNACER in practice a policy-gradient method.

7.2.2 Learning from Demonstration Data

From the user's perspective, performing RL from scratch will invariably result in unacceptable performance in the early learning stages. This problem can be mitigated by using an off-line corpus of *demonstration data* to bootstrap a policy. This data may come from a WoZ collection or from interactions between users and an existing policy. It can be used in three ways: **(A)**: Pre-train the model, **(B)**: Initialise a supervised replay buffer $\mathcal{P}_{sup}$, and **(C)**: a combination of the two.

(A) For model pre-training, the objective is to 'mimic' the response behaviour from the corpus. This phase is essentially standard SL. The input to the model is the dialogue belief state **b**, and the training objective for each sample is to minimise a joint cross-entropy loss $\mathcal{L}(\theta) = -\sum_k y_k \log(p_k)$ between action labels y and model predictions p, where the policy is parametrised by a set θ.

A policy trained by SL on a fixed dataset may not generalise well. In spoken dialogues, the noise levels may vary across conditions and thus significantly affect performance. Moreover, a policy trained using SL does not perform any long-term planning on the conversation. Nonetheless, supervised pre-training offers a good model starting point which can then be fine-tuned using RL.

(B) For supervised replay initialisation, the demonstration data is stored in a replay pool $\mathcal{P}_{sup}$ which is kept separate from the ER pool used for RL and is never over-written. At each RL update iteration, a small portion of the demonstration data $\mathcal{P}'_{sup}$ is sampled, and the supervised cross-entropy loss $\mathcal{L}(\theta)$ computed on this data is added to the RL objective $J(\theta)$. Also, an L2 regularisation loss $\|\cdot\|_2^2$ is applied to θ to help prevent it from over-fitting on the sampled demonstration dataset. The total loss to be minimised is then the following:

$$\mathcal{L}_{all}(\theta) = -J(\theta) + \lambda_1 \mathcal{L}(\theta; \mathcal{P}'_{sup}) + \lambda_2 \|\theta\|_2^2, \tag{7.11}$$

where λ's are weights. In this way, the RL policy is guided by the sampled demonstration data while learning to optimise the total return.

(C) The learned parameters of the pre-trained model in method A might distribute differently from the optimal RL policy, and this may cause some performance drop in early stages while learning an RL policy from this model. This can be alleviated by using the composite loss proposed in method B. A comparison between the three options is included in the experimental evaluation.

7.3 Experiments

The experiments described in this chapter utilised the software tool-kit PyDial (Ultes et al., 2017), which provides a platform for modular SDS. Similar to previous chapters the target application is a live telephone-based SDS providing restaurant information for the Cambridge (United Kingdom) area. The task is to learn a policy which manages the dialogue flow and delivers requested information to the user. The domain consists of approximately 100 venues, each with six slots out of which three can be used by the system to constrain the search (food-type, area and price-range) and three are system-informable properties (phone-number, address and postcode) available once a database entity has been found. The former three slots are also referred to as requestable slots.

The input for all models was the full dialogue belief state $\mathbf{b}$ of size 268 which includes the last system action and distributions over the user intention and the three requestable slots. The output includes 14 restricted dialogue actions determining the system intent at the semantic level. Table 7.1 shows the entire list of restricted system actions.

Table 7.1 The list of restricted system actions.

System Action	Description
`request_area`	system asks user to provide the value of area
`request_pricerange`	system asks user to provide the value of price range
`request_food`	system asks user to provide the value of food type
`confirm_area`	system asks user to confirm the value of area
`confirm_pricerange`	system asks user to confirm the value of price range
`confirm_food`	system asks user to confirm the value of food
`select_area`	system asks user to choose from two values of area
`select_pricerange`	system asks user to choose from two values of price range
`select_food`	system asks user to choose from two values of food
`inform`	system informs the slot-values with probability > threshold
`inform_byname`	system informs the name of an entity
`inform_alternatives`	system informs an alternative entity
`inform_requested`	system informs the value of the user-requested slots
`bye`	system says good bye

Combining the dialogue belief states and heuristic rules, the selected action is then mapped into a spoken response using a natural language generator.

7.3.1 Model Comparison

Two value-based methods are shown for comparison with the policy-based models described. For both, the policy is implicitly determined by the action-value (Q) function which estimates the expected total return when choosing action a given belief state $\mathbf{b}$ at time-step t. For an optimal policy π^*, the Q-function satisfies the *Bellman equation* (Bellman, 1954):

$$Q^*(\mathbf{b}_t, a_t) = \mathrm{E}_{\pi^*}\{r_t + \gamma \max_{a'} Q^*(\mathbf{b}_{t+1}, a')|\mathbf{b}_t, a_t\}. \tag{7.12}$$

Deep Q-Network (DQN)

The Deep Q-Network is a variant of the Q-learning algorithm whereby a neural network is used to approximate the Q-function. This suggests a sequential approximation in Equation 7.12 by minimising the loss:

$$L(w_t) = \mathbb{E}\left[(y_t \; - Q(\mathbf{b}_t, a_t; w_t))^2\right], \tag{7.13}$$

where $y_t = r_t + \gamma \max_{a'} Q(\mathbf{b}_{t+1}, a'; w_t^-)$ is the target to update the parameters w. Note that y_t is evaluated by a target network w^- which is updated less frequently than the network w to stabilise learning, and the expectation is over the tuples $(\mathbf{b}_t, a_t, r_{t+1}, \mathbf{b}_{t+1})$ sampled from the ER pool described in Section 7.2.1.

DQN often suffers from over-estimation of Q-values as the max operator is used to select an action as well as to evaluate it. Double DQN (DDQN) (Van Hasselt et al., 2016) is thus used to de-couple the action selection and Q-value estimation to achieve better performance.

GP-SARSA

GP-SARSA is a state-of-the-art value-based RL algorithm for dialogue modelling and has been shown in the previous chapters to be effective. It is appealing since it can learn from a small number of observations by exploiting the correlations defined by a *kernel function* and provides an uncertainty measure of its estimates. In this work, GP-SARSA with uncertainty estimate is used as the benchmark.

7.3.2 Reinforcement Learning from Scratch

The proposed models were first evaluated under 0% semantic error rate with an agenda-based simulator which generates user interactions at the semantic-level (Schatzmann et al., 2006). In this case, the user intent is perfectly captured in the dialogue belief state without noise.

The total return of each dialogue was set to $\mathbb{1}(\mathcal{D}) - 0.05 \times T$, where T is the dialogue length and $\mathbb{1}(\mathcal{D})$ is the success indicator for dialogue $\mathcal{D}$. The maximum dialogue length was set to 20 turns, and γ was 0.99. All deep RL models (A2C, TRACER, eNACER and DQN) contained two hidden layers of size 130 and 50. The Adam optimiser was used (Kingma and Ba, 2014) with an initial learning rate of 0.001. During training, an ϵ-greedy policy was used, which was initially set to 0.3 and annealed to 0.0 over 3500 training dialogues. For GP, a linear kernel was used.

The ER pool $\mathcal{P}$ size was 1000, and the mini-batch size was 64. Once an initial 192 samples had been collected, the model was updated after every 2 dialogues. Note that for DQN, each sample was a state transition $(\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1})$, whereas in A2C, TRACER and eNACER, each sample comprised the whole dialogue with all its state transitions. For eNACER, the natural gradient was computed to update the model weights of size ~ 42000. For TRACER, α was set to 0.02, and ξ was 0.01. Since the IS ratio has a high variance and can occasionally be extremely large, it was clipped between [0.8,1.0] to maintain stable training.

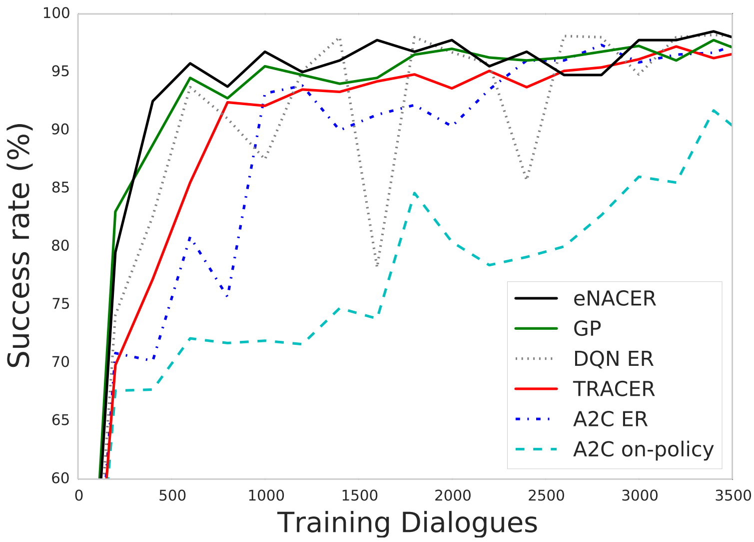
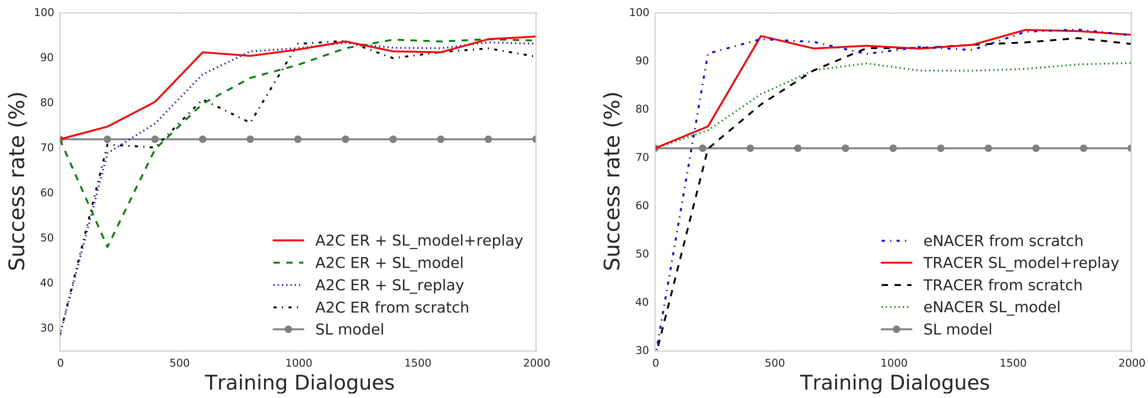


Fig. 7.2 The success rate learning curves of on-policy A2C, A2C with ER, TRACER, DQN with ER, GP, and eNACER in user simulation under noise-free condition.

Figure 7.2 shows the success rate learning curves of on-policy A2C, A2C with ER, TRACER, DQN with ER, GP and eNACER. All were tested with 600 dialogues after every 200 training dialogues. As reported in previous studies, the benchmark GP model learns quickly and is relatively stable. eNACER provides comparable performance. DQN also showed high sample-efficiency but with high instability at some points. This is because an iterative improvement in value space does not guarantee an improvement in policy space. Although comparably slower to learn, the difference between on-policy A2C and A2C with ER clearly demonstrates the sample-efficiency of re-using past samples in mini-batches. The enhancements incorporated into the TRACER algorithm do make this form of learning competitive although it still lags behind eNACER and GP-SARSA.

7.3.3 Learning from Demonstration Data

Regardless of the choice of model and learning algorithm, training a policy from scratch on-line will always result in a poor user experience until sufficient interactions have been experienced to allow acceptable behaviours to be learned.



(a) Learning for A2C ER. (b) Learning for TRACER and eNACER.

Fig. 7.3 Utilising demonstration data for improving RL learning speed.

As discussed in Section 7.2.2, an off-line corpus of demonstration data can potentially mitigate this problem. To test this, a corpus of 720 real user spoken dialogues in the Cambridge restaurant domain was utilised. The corpus was split in a 4:1:1 ratio for training, validation and testing. It contains interactions between real users recruited via the AMT service and a well-behaved SDS as described in Su et al. (2016b).

For A2C with ER and TRACER, the three ways of exploiting demonstration data in Section 7.2.2 were explored. The exploration parameter ϵ was also set to 0.3 and annealed to 0.0 over 2000 training dialogues. Since TRACER has similar patterns to A2C with ER, we first explored the impact of demonstration data on the A2C with ER results since it provides more headroom for identifying performance gains.

Figure 7.3a shows the different combinations of demonstration data using A2C with ER in noise-free conditions. The supervised pre-trained model (SL model) provides reasonable starting performance. The A2C ER model with supervised pre-training (A2C ER+SL_model) improves on this after only 400 dialogues while suffering initially. We hypothesise that the optimised SL pre-trained parameters distributed very differently to the optimal eNACER ER parameters. Also, the A2C ER model with SL replay (A2C ER+SL_replay) shows clearly how the use of a supervised replay buffer can accelerate learning from scratch. Moreover, when SL pre-training is combined with SL replay (A2C ER+SL_model+replay), it achieved the best result. In addition, λ_1 and λ_2 in Equation 7.11 were 10 and 0.01, respectively. In each policy update, 64 demonstration data were randomly sampled from the supervised replay pool $\mathcal{P}_{sup}$, which is the same number of RL samples selected from ER for A2C learning. Similar patterns emerge when utilising demonstration data to improve early learning in the TRACER and eNACER algorithms as shown in Figure 7.3b. However, in this case, eNACER is less able to exploit demonstration data since the training method is different from standard actor-critics. Hence, the supervised loss $\mathcal{L}$ cannot be directly incorporated into the RL objective J, as in Equation 7.11. One could optimise the model using $\mathcal{L}$ separately after every RL update. However, in our experiments, this did not yield improvement. Hence, only eNACER learning from a pre-trained SL model is reported here. Compared to eNACER learning from scratch, eNACER from SL model started with good performance but learned more slowly. Again, this may be because the optimised SL pre-trained parameters distributed very differently from the optimal eNACER parameters and led to sub-optimality. Overall, these results suggest that the proposed SL+RL framework to exploit demonstration data is effective in mitigating the cold start problem and TRACER provides the best solution regarding avoiding poor initial performance, rapid learning and competitive fully trained performance.

In addition to the noise-free performance, we also investigated the impact of noise on the TRACER algorithm. Figure 7.4 shows the results after training on 2000 dialogues via interaction with the user simulator under different semantic error rates. The random policy (white bars) uniformly sampled an action from the action set of size 14. This can be regarded as the average initial performance of any learning system. The figure shows that SL generates a robust model which can be further fine-tuned using RL over a wide range of error rates. It should be noted, however, that the drop-off in performance at high noise levels is more rapid than might be expected, comparing to the GP-SARSA.

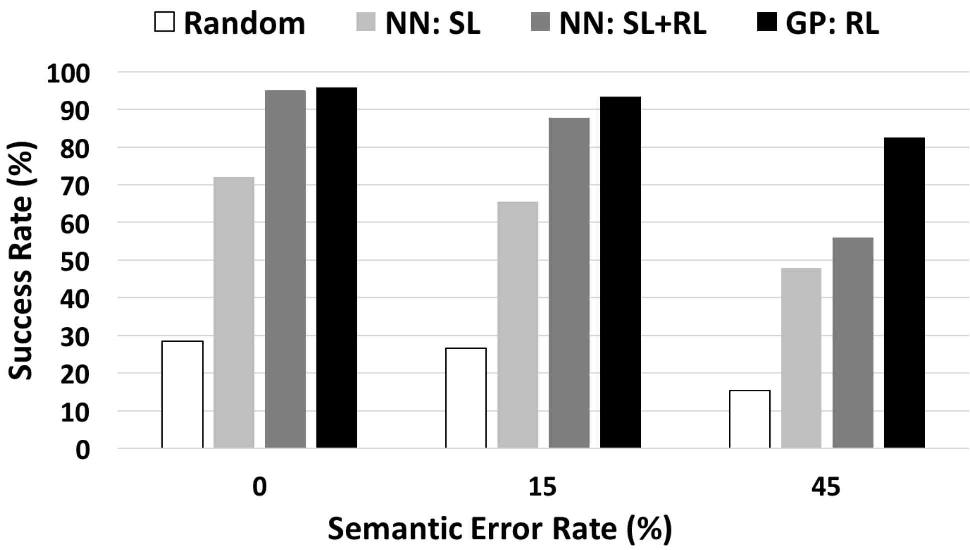


Fig. 7.4 The success rate of TRACER for a random policy, policy trained with corpus data (NN:SL), further improved via RL (NN:SL+RL), and GP RL (GP:RL), respectively, in user simulation under various semantic error rates.

To examine the learning details, Figure 7.5 illustrates the success rate learning curve of TRACER under SERs 15% and 45%. The models learned from the SL-trained models, which are the cases of no RL training dialogues in the figure. We can clearly see that the proposed model learned fast up to 200 training dialogues in both SERs but struggled to improve much afterwards It is suspected that deep architectures are prone to overfitting and in consequence do not handle well uncertainty in user behaviour. Overall, these outcomes validate the benefit of the proposed two-phased approach where the system can be effectively pre-trained using corpus data and then be further refined via user interactions.

7.4 Summary

This chapter has presented two compatible approaches to addressing the problem of slow learning and poor initial performance in deep RL algorithms. First, TRACER and eNACER were presented, and these have been shown to be more sample-efficient than other deep RL models and broadly competitive with GP-SARSA.

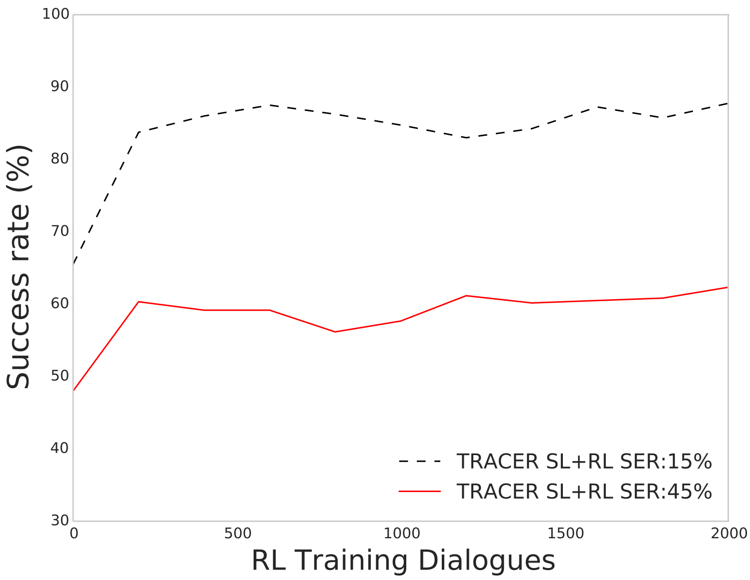


Fig. 7.5 The success rate learning curve of TRACER trained with SL+RL data in user simulation under semantic error rates 15% and 45%.

Second, it has been shown that demonstration data can be utilised to mitigate poor performance in the early stages of learning. To this end, two methods for using off-line corpus data were presented: simple pre-training using SL, and using the corpus data in a replay buffer. These were particularly effective when used with TRACER which provided the best overall performance.

Experimental results were also presented for mismatched environments, and again, TRACER demonstrated the ability to avoid poor initial performance when trained only on the demonstration corpus, yet it still improves substantially with subsequent RL. However, performance still falls off rather rapidly in noise compared to GP-SARSA, as the uncertainty estimates are not handled well by neural networks architectures.

Finally, it should be emphasised that while this chapter has focussed on the early stages of learning, a domain where GP-SARSA provides a benchmark that is hard to beat, the potential of deep RL is its scalability to exploit on-line learning with very large user populations since the model size is fixed and is independent to the collected samples.

CHAPTER 8

Conclusions

8.1 Conclusions

Reinforcement learning approaches provide a general purpose framework for decision making, which is beneficial for automating dialogue policy design without many hand-crafted rules. This dissertation addresses two main challenges of RL-based dialogues: the practical design of the learning objective and the training speed of the system.

An appropriate RL objective in dialogue is not readily apparent since it is determined by the human user's own internal goal, which is hard to measure. To address this issue, this thesis first describes an RNN-based reward estimator, which is trained from off-line data, for estimating task success during subsequent on-line dialogue policy learning. Furthermore, a joint on-line dialogue policy and reward learning framework is also proposed, where the reward is modelled as a GP with active learning to handle noise in user ratings and minimise user intrusion. Both off-line and on-line methods have been shown to achieve practical policy learning in dialogue applications, while the latter provides a more general system able to learn jointly directly from users.

In dialogue systems, the on-line RL scenario has two common issues: slow learning speed and poor performance in the early training stages. These problems have a negative impact on user experience. Reward shaping, a light-weighted and effective concept, is exploited to enrich the original sparse reward signal to enhance the policy learning speed. Moreover, because of their potential to scale to large real-world tasks, deep neural network policy models are also investigated for learning a dialogue agent. Two sample-efficient algorithms, trust region actor-critic with experience replay (TRACER) and episodic natural actor-critic with experience replay (eNACER) are introduced to address the slow learning of gradient-based methods. Furthermore, a corpus of demonstration data is utilised to pre-train the models prior to on-line RL to handle the cold start problem. Combining the above methods, a practical approach is suggested to effectively learn deep RL-based dialogue policies in a task-oriented information seeking domain.

Overall, this thesis contributes to advance the SDS that can continuously learn and adjust its policy on-line with real users.

8.2 Limitation and Future Work

The reward estimators proposed in the first half of the thesis are closely related to dialogue evaluation (Dodge et al., 2015; Liu et al., 2016; Lowe et al., 2017). Although in task-oriented slot-filling scenarios, success information is the prominent feature, this is only one aspect of user satisfaction. It is thus worth investing other dimensions that define the dialogue quality.

Another natural extension to the work here is on deep RL-based dialogue policy. Unlike Bayesian methods such as GPs which have uncertainty measurement, the exploration strategy used in the neural network-based models is always ϵ-greedy, where all actions are served equally; thus, the model learns slowly. One possible way is to incorporate uncertainty measurement into the neural networks (Blundell et al., 2015; Gal and Ghahramani, 2015; Osband et al., 2016). Tegho et al. (2017) has shown positive results for enhancing the sample-efficiency of DQN models.

Current dialogue policies mainly operate on dialogue acts (Stolcke et al., 2000), of which there are typically around 10–20. However, these dialogue acts are often hand-crafted, and thus scalability remains a big issue because the choice of dialogue act highly constrains the potential output response. A recent attempt has been made to learn latent dialogue acts (Wen et al., 2017b) and this may help generate a more diverse system response.

Despite the promising results of the methods explored in this thesis, there are several issues remaining in the current modular SDSs. While the pipeline architecture enables us to diagnose and improve individual components, the improvement of a single module may not directly boost the performance of the downstream units. Recent work on end-to-end text-based task-oriented dialogue systems (Bordes et al., 2017; Eric and Manning, 2017; Wen et al., 2017a; Yang et al., 2017), on the other hand, aims to jointly learn multiple components together and do not factorise the model into intermediate states; thus, no labels are needed. These methods potentially speed up the development process of the entire dialogue systems and are worth further investigation.

Apart from pure natural language-based dialogues, researchers in the computer vision community have tried to combine images with text-based dialogues (Das et al., 2016, 2017). Essentially, the visual inputs provide additional rich information about the environment on top of the dialogue exchange and thus help achieve good performance. More joint work between language and vision are expected to appear in the near future.

Although learning from real users is desirable, it is often costly. On the other hand, user simulation (Schatzmann et al., 2006) offers a useful testbed for evaluating the newly proposed methods, where hundreds of thousands of interactions can be done with little cost. In addition, in the RL setting, the user is the environment the system is interacting with. Hence, learning a user model is equivalent to learning the environmental transition function (transition probability). Combining this with the subsequent dialogue policy learning, the entire framework can be considered as a model-based RL task. As opposed to model-free methods which require many samples to approximate the policy, model-based approaches may be far more sample-efficient.

Acronyms

A2C Advantage Actor-Critic. xviii, 94–96, 98, 103–105

AMT Amazon mechanical turk. 57, 67, 88, 104

ASR automatic speech recognition. 10, 12, 14, 15, 19, 21, 23

BUDS Bayesian update of dialogue state. 13, 51

CFG context-free grammars. 11, 12

CRF conditional random fields. 12, 13

DBN dynamic Bayesian network. 12, 13, 51

DiaAct dialogue act. 11

DQN Deep Q-Network. xviii, 102–104

DST dialogue state tracking. 12–15, 19, 23, 24, 37

eNACER episodic natural actor-critic with experience replay. xviii, 92, 94, 99, 103–107, 110

ER experience replay. 92, 95, 102

ES evolutionary strategies. 28, 29

GMM Gaussian mixture model. 10

GP Gaussian process. xviii, 5, 6, 28, 49, 51, 52, 60, 65, 78, 82, 91–93, 102–104, 106, 107, 109, 110

GRU gated recurrent unit. xvii, 48, 82–89

HMM hidden Markov model. 10, 16, 51

IRL inverse reinforcement learning. 39, 40

LM language model. 10, 15

LSTM long short-term memory. xvi, xvii, 48, 62, 63, 82–86, 89

MC Monte-Carlo. 35

MDP Markov decision process. 15, 20, 21, 25, 41

MFCCs Mel frequency cepstral coefficients. 10

NLG natural language generation. 15, 23, 40, 46, 52

NLP natural language processing. 46, 83

POMDP partially observable Markov decision process. 15, 20, 21, 23–28, 33, 34, 38

RL reinforcement learning. xv, xviii, 4–7, 14, 16, 17, 19, 20, 24, 25, 28, 32–38, 40, 42, 45, 52, 77, 79, 90–93, 95–97, 99, 100, 102, 103, 105–107, 109–112

RNN recurrent neural network. xv, xvii, xix, 6, 10, 46, 48, 49, 55, 58, 61–63, 68, 73, 75, 78, 81–87, 89, 90, 109

SDS spoken dialogue system. 1–3, 5, 9, 12, 19, 21–24, 32–34, 37–40, 42, 45, 51, 52, 58, 65, 67, 91, 93, 101, 104, 110, 111

SGD stochastic gradient descent. 54, 64, 85

SLU spoken language understanding. 11–14, 19, 21–23, 37, 46

SS speech synthesis. 16

SVM support vector machine. 12

TD temporal difference. 35, 95

TRACER trust region actor-critic with experience replay. xviii, 92, 94, 98, 99, 103–107, 110

TTS text-to-speech. 23

References

Amari, S.-I. (1998). Natural gradient works efficiently in learning. In *Neural computation*, volume 10, pages 251–276. MIT Press.

Antos, A., Szepesvári, C., and Munos, R. (2008). Fitted q-iteration in continuous action-space mdps. In *Proc of NIPS*, pages 9–16.

Asadi, K. and Williams, J. D. (2016). Sample-efficient deep reinforcement learning for dialog control. In *arXiv preprint arXiv:1612.06000*.

Asri, L. E., He, J., and Suleman, K. (2016). A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070*.

Asri, L. E., Laroche, R., and Pietquin, O. (2014). Task completion transfer learning for reward inference. In *Proc of MLIS*.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *Proc of ICASSP*.

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proc of ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS Workshop.

Bellman, R. (1954). The theory of dynamic programming. Technical report, DTIC Document.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. volume 5. IEEE.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *Proc of ICML*.

Bordes, A., Boureau, Y.-L., and Weston, J. (2017). Learning end-to-end goal-oriented dialog. In *Proc of ICLR*.

Boularias, A., Chinaei, H. R., and Chaib-draa, B. (2010). Learning the reward model of dialogue pomdps from data. In *NIPS Workshop on Machine Learning for Assistive Techniques*.

Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. In *arXiv preprint arXiv:1012.2599*.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep gaussian processes for regression using approximate expectation propagation. *Proc of ICML*.

Buşoniu, L., Ernst, D., De Schutter, B., and Babuška, R. (2011). Approximate reinforcement learning: An overview. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 1–8. IEEE.

Casanueva, I., Budzianowski, P., Su, P.-H., Mrkšić, N., Wen, T.-H., Ultes, S., Rojas-Barahona, L., Young, S., and Gašić, M. (2017). A benchmarking environment for reinforcement learning based task oriented dialogue management. *NIPS Deep Reinforcement Learning Symposium*.

Casanueva, I. n., Hain, T., Christensen, H., Marxer, R., and Green, P. (2015). Knowledge transfer between speakers for personalised dialogue management. In *Proc of SIGDIAL*.

Chai, J. Y., Kambhatla, N., and Zadrozny, W. (2001). Natural language sales assistant-a web-based dialog system for online sales. In *IAAI*.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proc of ICASSP*.

Chandramohan, S., Geist, M., Lefevre, F., and Pietquin, O. (2011). User simulation in dialogue systems using inverse reinforcement learning. In *Proc of Interspeech*.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005(598-603):18.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proc of EMNLP*.

Chen, L., Su, P.-H., and Gašic, M. (2015). Hyper-parameter optimisation of gaussian process reinforcement learning for statistical dialogue management. In *Proc of SIGDIAL*.

Chen, X., Liu, X., Wang, Y., Gales, M. J., and Woodland, P. C. (2016). Efficient training and evaluation of recurrent neural network language models for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2146–2157.

Chen, Y.-N., Wang, W. Y., and Rudnicky, A. I. (2013). Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 120–125. IEEE.

Cheng, W., Fürnkranz, J., Hüllermeier, E., and Park, S.-H. (2011). Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *Machine learning and knowledge discovery in databases*. Springer.

Cho, K., Gulcehre, B. v. M. C., Bahdanau, D., Schwenk, F. B. H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Dai, P., Lin, C. H., Weld, D. S., et al. (2013). Pomdp-based control of workflows for crowdsourcing. In *Artificial Intelligence*, volume 202. Elsevier.

Daniel, C., Viering, M., Metz, J., Kroemer, O., and Peters, J. (2014). Active reward learning. In *Proc of RSS*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2016). Visual dialog. *arXiv preprint arXiv:1611.08669*.

Das, A., Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Learning cooperative visual dialog agents with deep reinforcement learning. *ICCV*.

Daubigney, L., Geist, M., Chandramohan, S., and Pietquin, O. (2012a). A comprehensive reinforcement learning framework for dialogue management optimisation. volume 6.

Daubigney, L., Geist, M., and Pietquin, O. (2012b). Off-policy learning in large-scale pomdp-based dialogue systems. In *IEEE ICASSP*, pages 4989–4992.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142.

Deng, L., Hinton, G., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE.

Dewey, D. (2014). Reinforcement learning and the reward engineering principle. In *Proc of AAAI Spring Symposium Series*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2016). End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. (2015). Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Eck, A., Soh, L.-K., Devlin, S., and Kudenko, D. (2015). Potential-based reward shaping for finite horizon online pomdp planning. pages 1–43.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User modeling for spoken dialogue system evaluation. In *IEEE ASRU*, pages 80–87.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Engel, Y. (2005). *Algorithms and representations for reinforcement learning*. PhD Thesis.

Eric, M. and Manning, C. D. (2017). Key-value retrieval networks for task-oriented dialogue. *Proc of SIGDIAL*.

Fatemi, M., Asri, L. E., Schulz, H., He, J., and Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. In *Proc of SIGDIAL*.

Ferreira, E. and Lefèvre, F. (2013). Social signal and user adaptation in reinforcement learning-based dialogue management. In *Proceedings of the 2Nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*, MLIS '13, pages 61–69, New York, NY, USA. ACM.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*.

Gales, M. and Young, S. (2008). The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304.

Gandhe, S. and Traum, D. (2008). An evaluation understudy for dialogue coherence models. In *Proc of SIGDIAL*, pages 172–181. Association for Computational Linguistics.

Gašic, M. (2011). *Statistical dialogue modelling*. PhD thesis, University of Cambridge.

Gašić, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. (2013a). Pomdp-based dialogue manager adaptation to extended domains. In *Proc of SIGDIAL*.

Gašić, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. J. (2013b). On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *Proc of ICASSP*.

Gašić, M., Jurcicek, F., Thomson, B., Yu, K., and Young, S. (2011). On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 312–317.

Gašić, M., Kim, D., Tsiakoulis, P., Breslin, C., Henderson, M., Szummer, M., Thomson, B., and Young, S. J. (2014). Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. In *Proc of Interspeech*.

Gašić, M. and Young, S. (2014). Gaussian processes for POMDP-based dialogue manager optimisation. *TASLP*, 22.

Gasic, M. and Young, S. (2014). Gaussian processes for POMDP-based dialogue manager optimization. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(1):28–40.

Gašić, M., Breslin, C., Henderson, M., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. (2013). On-line policy optimisation of Bayesian Dialogue Systems by human interaction.

Georgila, K., Tsopanoglou, A., Fakotakis, N., and Kokkinakis, G. (1998). An integrated dialogue system for the automation of call centre services. In *ISCLP*.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483.

Goldstein, J., Kantrowitz, M., Mittal, V., and Carbonell, J. (1999). Summarizing text documents: sentence selection and evaluation metrics. In *Proc of SIGIR*, pages 121–128.

Graves, A., Jaitly, N., and Mohamed, A.-r. (2013a). Hybrid speech recognition with deep bidirectional lstm. In *IEEE ASRU*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013b). Speech recognition with deep recurrent neural networks. In *IEEE ICASSP*, pages 6645–6649.

Greensmith, E., Bartlett, P. L., and Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530.

Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2016). Q-prop: Sample-efficient policy gradient with an off-policy critic. *Proc of ICLR*.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Harispe, S., Sánchez, D., Ranwez, S., Janaqi, S., and Montmain, J. (2014). A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain. *Journal of biomedical informatics*, 48:38–53.

He, Y. and Young, S. (2006). Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3):262–275.

Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. In *Computational Linguistics*, volume 34, pages 487–511. MIT Press.

Henderson, M., Gašić, M., Thomson, B., Tsiakoulis, P., Yu, K., and Young, S. (2012). Discriminative spoken language understanding using word confusion networks. In *IEEE SLT*.

Henderson, M., Thomson, B., and Williams, J. (2014a). The second dialog state tracking challenge. In *Proc of SIGDIAL*, volume 263.

Henderson, M., Thomson, B., and Williams, J. D. (2014b). The third dialog state tracking challenge. In *IEEE SLT*, pages 324–329.

Henderson, M., Thomson, B., and Young, S. J. (2014c). Word-based Dialog State Tracking with Recurrent Neural Networks. In *Proc of SIGdial*.

Hensman, J., Fusi, N., Andrade, R., Durrande, N., Saul, A., Zwiessele, M., and Lawrence, N. D. (2012). GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. *Proc of NIPS*.

Hester, T., Vecerik, M., Pietquin, O., andTom Schaul, M. L., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z., and Gruslys, A. (2017). Learning from demonstrations for real world reinforcement learning. In *arXiv:1704.03732*.

Higashinaka, R., Meguro, T., Imamura, K., Sugiyama, H., Makino, T., and Matsuo, Y. (2014). Evaluating coherence in open domain conversational systems. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hirschman, L. and Thompson, H. S. (1997). Overview of evaluation in speech and natural language processing.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. volume 9. MIT Press.

Holmes, W. (2001). *Speech synthesis and recognition*. CRC press.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. (2017). Controllable text generation. *arXiv preprint arXiv:1703.00955*.

Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE.

Janarthanam, S., Hastie, H., Lemon, O., and Liu, X. (2011). The day after the day after tomorrow?: a machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *Proc of SIGDIAL*, pages 142–151. Association for Computational Linguistics.

Jebara, T., Kondor, R., and Howard, A. (2004). Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844.

Jeong, M. and Lee, G. G. (2008). Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *ICML*.

Jung, S., Lee, C., Kim, K., Jeong, M., and Lee, G. G. (2009). Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech & Language*, 23(4):479–509.

Jurafsky, D. and Martin, J. H. (2008). Speech and language processing.

Jurčíček, F. (2012). Reinforcement learning for spoken dialogue systems using off-policy natural gradient method. In *IEEE SLT*, pages 7–12.

Jurčíček, F., Keizer, S., Gašić, M., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2011). Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk. *Proc of Interspeech*.

Kadlec, R., Libovický, J., Macek, J., and Kleindienst, J. (2014). Ibm?s belief tracker: Results on dialog state tracking challenge datasets.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences.

Karpathy, A. and Fei-Fei, L. (2014). Deep visual-semantic alignments for generating image descriptions. volume abs/1412.2306.

Kate, R. J. and Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proc of ACL*.

Keizer, S., Gašić, M., Jurčíček, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010a). Parameter estimation for agenda-based user simulation. In *Proc of SIGDIAL*, pages 116–123. Association for Computational Linguistics.

Keizer, S., Gašić, M., Jurcicek, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010b). *Proc of SIGDIAL*, chapter Parameter estimation for agenda-based user simulation.

Kim, D., Breslin, C., Tsiakoulis, P., Henderson, M., and Young, S. J. (2014). Inverse reinforcement learning for micro-turn management. In *IEEE SLT*.

Kim, S., D?Haro, L. F., Banchs, R. E., Williams, J. D., and Henderson, M. (2017). The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449.

Kim, S., D?Haro, L. F., Banchs, R. E., Williams, J. D., Henderson, M., and Yoshino, K. (2016). The fifth dialog state tracking challenge. In *IEEE SLT*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*.

Konda, V. R. and Tsitsiklis, J. N. (1999). Actor-critic algorithms. In *Proc of NIPS*, volume 13, pages 1008–1014.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in ccg grammar induction for semantic parsing. In *Proc of EMNLP*, pages 1512–1523.

Lafferty, J., McCallum, A., Pereira, F., et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Larsen, L. (2003). Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In *Automatic Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop on*, pages 209–214.

Larsson, S. and Traum, D. R. (2000). Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3&4):323–340.

Lee, S. and Eskenazi, M. (2012). POMDP-based let's go system for spoken dialog challenge. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 61–66. IEEE.

Lee, S. and Eskenazi, M. (2013). Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proc of SIGDIAL*.

Lemon, O., Georgila, K., and Henderson, J. (2006). Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the talk towninfo evaluation. In *IEEE SLT*, pages 178–181.

Lemon, O. and Pietquin, O. (2007). Machine learning for spoken dialogue systems. In *Proc of Interspeech*.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using markov decision process for learning dialogue strategies. In *IEEE ICASSP*.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. volume 8, pages 11–23.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *NIPS*.

Li, J., Monroe, W., Ritter, A., and Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. In *Proc of EMNLP*.

Li, X., Chen, Y.-N., Li, L., Gao, J., , and Celikyilmaz, A. (2017). Investigation of language understanding impact for reinforcement learning based dialogue systems. In *arXiv: 1703.07055*.

Lin, C. H., Weld, D. S., et al. (2014). To re (label), or not to re (label). In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. In *Machine learning*, volume 8, pages 293–321.

Ling, Z.-H., Kang, S.-Y., Zen, H., Senior, A., Schuster, M., Qian, X.-J., Meng, H. M., and Deng, L. (2015). Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 32(3):35–52.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y., and Pineau, J. (2017). Towards an automatic turing test: Learning to evaluate dialogue responses. *Proc of ACL*.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Mairesse, F. and Young, S. (2014). Stochastic language generation in dialogue using factored language models. *Computational Linguistics*.

Mangu, L., Brill, E., and Stolcke, A. (2000). Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Martens, J. (2014). New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*.

Mataric, M. J. (1994). Reward functions for accelerated learning.

McTear, M. F. (1998). Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. *development*, 5(7).

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2015a). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(3):530–539.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2015b). Using recurrent neural networks for slot filling in spoken language understanding. volume 23, pages 530–539. IEEE.

Meuleau, N., Peshkin, L., Kaelbling, L. P., and Kim, K.-E. (2000). Off-policy policy search. In *Technical report, MIT AI Lab*.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., and Cernockỳ, J. (2011a). Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, number s 1, pages 605–608.

Mikolov, T., Karafiat, M., and Burget, L. (2010). Recurrent neural network based language model. In *Proc of INTERSPEECH*.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J. H., and Khudanpur, S. (2011b). Extensions of recurrent neural network language model. In *ICASSP*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NIPS*.

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Proc of NIPS*, pages 2204–2212.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Gašić, M., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2015). Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *Proc of ACL*.

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Wen, T.-H., and Young, S. (2017). Neural Belief Tracker: Data-driven dialogue state tracking. In *Proc of ACL*.

Murphy, K. P. (2000). A survey of POMDP solution techniques. *environment*, 2.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc of ICML*.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary gaussian process classification. In *JMLR*, volume 9.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Oh, A. H. and Rudnicky, A. I. (2000). Stochastic language generation for spoken dialogue systems. In *Proc of ANLP/NAACL Workshop on Conversational systems*, pages 27–32.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. In *Proc of NIPS*.

Paek, T. and Pieraccini, R. (2008). Automating spoken dialogue management design using machine learning: An industry perspective. volume 50. Elsevier.

Papadimitriou, C. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450.

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

Perez, J. and Liu, F. (2016). Dialog state tracking, a machine reading approach using memory network. *arXiv preprint arXiv:1606.04052*.

Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In *IEEE RSJ*.

Pieraccini, R. and Huerta, J. (2005). Where do we go from here? research and commercial spoken dialog systems. In *Proc of SIGDIAL*.

Pietquin, O. (2013). Inverse reinforcement learning for interactive systems. In *Proceedings of the 2nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*, pages 71–75. ACM.

Pietquin, O. and Dutoit, T. (2006). A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599.

Pietquin, O. and Hastie, H. (2013). A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(01):59–73.

Pineau, J., Gordon, G., Thrun, S., et al. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032.

Precup, D., Sutton, R. S., and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proc of ICML*.

Price, P. (1990). Evaluation of spoken language systems: The atis domain. In *Proc of HLT*.

Randløv, J. and Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proc of ICML*.

Rapaport, W. J. (1986). Logical foundations for belief representation. *Cognitive Science*, 10(4):371–422.

Rasmussen, C. E. and Williams, C. (2006). Gaussian processes for machine learning. MIT Press.

Raymond, C. and Riccardi, G. (2007). Generative and discriminative algorithms for spoken language understanding. In *Proc of INTERSPEECH*.

Ren, H., Xu, W., Zhang, Y., and Yan, Y. (2013). Dialog state tracking using conditional random fields. In *Proc of SIGDIAL*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Rieser, V. and Lemon, O. (2006). Using logistic regression to initialise reinforcement-learning-based dialogue systems. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 190–193. IEEE.

Rieser, V. and Lemon, O. (2010). Natural language generation as planning under uncertainty for spoken dialogue systems. In *Empirical methods in natural language generation*, pages 105–120. Springer.

Rieser, V. and Lemon, O. (2011). Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. In *Computational Linguistics*, volume 37. MIT Press.

Rojas Barahona, L. M. and Cerisara, C. (2014). Bayesian Inverse Reinforcement Learning for Modeling Conversational Agents in a Virtual Environment. In *Conference on Intelligent Text Processing and Computational Linguistics*.

Rubinstein, R. Y. and Kroese, D. P. (2013). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media.

Russell, S. (1998). Learning agents for uncertain environments. In *Proc of COLT*.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Schatzmann, J., Georgila, K., and Young, S. (2005). Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proc of SIGDIAL*.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. (2007a). Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

Schatzmann, J., Thomson, B., and Young, S. (2007b). Error simulation for training statistical dialogue systems. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 526–531. IEEE.

Schatzmann, J., Weilhammer, K., Stuttle, M., and Young, S. (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126.

Schatzmann, J. and Young, S. (2009). The hidden agenda user simulation model. volume 17, pages 733–747. IEEE.

Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc of HLT*, pages 12–19.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. In *Proc of ICML*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *Proc of ICLR*.

Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Settles, B. (2010). Active learning literature survey. In *Computer Sciences Technical Report 1648*. University of Wisconsin, Madison.

Shani, G., Pineau, J., and Kaplow, R. (2013). A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. In *Nature*, volume 529, pages 484–489. Nature Publishing Group.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. (2000). Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pages 956–962.

Stent, A., Marge, M., and Singhai, M. (2005). Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016a). Continuously learning neural dialogue management. In *arXiv preprint arXiv:1606.02689*.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2017). Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proc of SIGDIAL*.

Su, P.-H., Gašić, M., and Young, S. (2018). Reward estimation for dialogue policy optimisation. *Computer Speech and Language*.

Su, P.-H., Gašić, M., Mrkšić, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016b). On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proc of ACL*.

Su, P.-H., Vandyke, D., Gašić, M., Kim, D., Mrkšić, N., Wen, T.-H., and Young, S. (2015a). Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Proc of INTERSPEECH*.

Su, P.-H., Vandyke, D., Gašić, M., Mrkšić, N., Wen, T.-H., and Young, S. (2015b). Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. In *Proc of SIGDIAL*.

Su, P.-h., Wang, Y.-B., Yu, T.-h., and Lee, L.-s. (2013). A dialogue game framework with personalized training using reinforcement learning for computer-assisted language learning. In *ICASSP*.

Su, P.-H., Wu, C.-H., and Lee, L.-S. (2015c). A recursive dialogue game for personalized computer-aided pronunciation training. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(1):127–141.

Sugiyama, H., Meguro, T., and Minami, Y. (2012). Preference-learning based inverse reinforcement learning for dialog control. In *Proc of Interspeech*.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *INTERSPEECH*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proc of NIPS*, pages 3104–3112.

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst.

Sutton, R. S. and Barto, A. G. (1999). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y., et al. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Proc of NIPS*.

Sutton, S., Novick, D. G., Cole, R., Vermeulen, P., de Villiers, J., Schalkwyk, J., and Fanty, M. (1996). Building 10,000 spoken dialogue systems. In *Proc of ICSLP*, volume 2, pages 709–712.

Tegho, C., Budzianowski, P., and Gašić, M. (2017). Uncertainty estimates for efficient neural network-based dialogue policy optimisation. *ICASSP*.

Thomson, B., Schatzmann, J., and Young, S. (2008). Bayesian update of dialogue state for robust dialogue systems. In *Proc of ICASSP*.

Thomson, B. and Young, S. (2010). Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. volume 24, pages 562–588.

Tsiakoulis, P., Breslin, C., Gasic, M., Henderson, M., Kim, D., Szummer, M., Thomson, B., and Young, S. (2014a). Dialogue context sensitive hmm-based speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2554–2558. IEEE.

Tsiakoulis, P., Breslin, C., Gašić, M., Henderson, M., Kim, D., and Young, S. (2014b). Dialogue context sensitive speech synthesis using factorized decision trees. In *Proc of INTERSPEECH*.

Tur, G. and Deng, L. (2011). *Intent Determination and Spoken Utterance Classification*, chapter 4, pages 81–104.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proc of ACL*.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

Ultes, S., Barahona, L. M. R., Su, P.-H., Vandyke, D., Kim, D., Casanueva, I., Budzianowski, P., Mrkšić, N., Wen, T.-H., Gašić, M., and Young, S. (2017). CU-PyDial: A Multi-domain Statistical Dialogue System Toolkit. In *ACL Demo*.

Ultes, S. and Minker, W. (2015). Quality-adaptive spoken dialogue initiative selection and implications on reward modelling. In *Proc of SIGDIAL*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. In *JMLR*, volume 9, page 85.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proc of AAAI*.

Vandyke, D., Su, P.-H., Gašić, M., Mrkšić, N., Wen, T.-H., and Young, S. (2015). Multi-domain dialogue success classifiers for policy training. In *IEEE ASRU*.

Vinyals, O. and Le, Q. (2015). A neural conversational model. In *arXiv preprint arXiv:1506.05869*.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1997). PARADISE: A framework for evaluating spoken dialogue agents. In *Proc of EACL*.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2017). Sample efficient actor-critic with experience replay. In *Proc of ICLR*.

Wang, Z. and Lemon, O. (2013). A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proc of SIGDIAL*.

Ward, W. (1994). Extracting information in spontaneous speech. In *Third International Conference on Spoken Language Processing*.

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

Wen, T.-H., Gašić, M., Mrkšić, N., M. Rojas-Barahona, L., Su, P.-H., Ultes, S., Vandyke, D., and Young, S. (2017a). A network-based end-to-end trainable task-oriented dialogue system. In *Proc of EACL*.

Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Wen, T.-H., Miao, Y., Blunsom, P., and Young, S. (2017b). Latent intention dialogue models. *Proc of ICML*.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. (2014). Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980.

Williams, J., Raux, A., Ramachandran, D., and Black, A. (2013). The dialog state tracking challenge. In *Proc of SIGDIAL*, pages 404–413.

Williams, J. D. (2014). Web-style ranking and slu combination for dialog state tracking. In *Proc of SIGDIAL*.

Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL*.

Williams, J. D. and Young, S. (2007a). Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Williams, J. D. and Young, S. (2007b). Partially observable Markov decision processes for spoken dialog systems. volume 21, pages 393–422.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning*, volume 8, pages 229–256. Springer.

Xu, P. and Sarikaya, R. (2013). Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.

Yang, X., Chen, Y.-N., Hakkani-Tür, D., Crook, P., Li, X., Gao, J., and Deng, L. (2017). End-to-end joint learning of natural language understanding and dialogue manager. In *IEEE ICASSP*, pages 5690–5694.

Yang, Z., Levow, G., and Meng, H. (2012). Predicting user satisfaction in spoken dialog system evaluation with collaborative filtering. volume 6, pages 971–981. IEEE.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *IEEE SLT*.

Young, S. (2009). Cued standard dialogue acts.

Young, S., Gasic, M., Thomson, B., and Williams, J. (2013a). POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Young, S., Gašic, M., Thomson, B., and Williams, J. (2013b). Pomdp-based statistical spoken dialogue systems: a review. In *Proc of IEEE*, volume 99, pages 1–20.

Young, S., Schatzmann, J., Weilhammer, K., and Ye, H. (2007). The hidden information state approach to dialog management. In *Proc of ICASSP*, volume 4.

Young, S. J. (2002). Talking to machines (statistically speaking). In *Proc of INTERSPEECH*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: sequence generative adversarial nets with policy gradient. In *Proc of AAAI*.

Zelle, J. M. and Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proc of AAAI*.

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., and Szczepaniak, P. (2016). Fast, compact, and high quality lstm-rnn based statistical parametric speech synthesizers for mobile devices. *arXiv preprint arXiv:1606.06061*.

Zen, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *Proc of ICASSP*.

Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064.

Zettlemoyer, L. S. and Collins, M. (2012). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

Zhang, C. and Chaudhuri, K. (2015). Active learning from weak and strong labelers. volume abs/1510.02847.

Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

Zielke, M. A., Evans, M. J., Dufour, F., Christopher, T. V., Donahue, J. K., Johnson, P., Jennings, E. B., Friedman, B. S., Ounekeo, P. L., and Flores, R. (2009). Serious games for immersive cultural training: Creating a living world. *IEEE computer graphics and applications*, 29(2):49–60.

Zue, V., Seneff, S., Glass, J. R., Polifroni, J., Pao, C., Hazen, T. J., and Hetherington, L. (2000). Juplter: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 8(1):85–96.

APPENDIX A

Dialogue Act Definitions

Dialogue acts offer a representation in semantic level for the user's or the system's utterance. The CUED dialogue act taxonomy (Young, 2009) is adopted in the entire thesis. A dialogue act is represented as the combination of a dialogue act type followed by a sequence of slot-value pairs (optional):

```
DiaActType(s1=v1, s2=v2).
```

The `DiaActType` is the type of the dialogue act, such as inform, request, or confirm. The slot-value pairs `s=v` are the contents of the utterance, where the `s` or `v` can be null. Examples are `area=north`, `phone=` and `=dontcare`. More details are described in Table A.1, where the CUED dialogue act definitions are presented.

Table A.1 A list of CUED dialogue acts.

Dialogue Act	System	User	Description
hello()	✓	✓	start the dialogue
hello(a=x,b=y, . . .)	×	✓	start the dialogue with information a=x, b=y, ...
silence()	×	✓	the user is silent
thankyou()	×	✓	implicit positive answer from the user
ack()	×	✓	back-channel e.g. uh huh, ok, etc
bye()	✓	✓	end the dialogue
hangup()	×	✓	user hangs up
inform(a=x, b=y, . . .)	✓	✓	give information a=x, b=y, ...
inform(name=none)	✓	×	inform that no matched entity is found
inform(a!=x, . . .)	×	✓	inform that a is not equal to x
inform(a=dontcare, . . .)	×	✓	the user does not care about the value of a
request(a)	✓	✓	request value of a
request(a, b=x, . . .)	✓	✓	request value of a given b=x,...
reqalts()	×	✓	request an alternative solution
reqalts(a=x, . . .)	×	✓	request an alternative solution with a=x,...
reqalts(a=dontcare, . . .)	×	✓	request an alternative solution relaxing constraint a
reqmore()	✓	×	inquire if user wants anything more
reqmore(a=dontcare)	✓	×	inquire if user would like to relax a
reqmore()	×	✓	request more information about the current solution
reqmore(a=x,b=y, . . .)	×	✓	request more information given a=x, b=y, ...
confirm(a=x,b=y, . . .)	✓	✓	confirm a=x, b=y, ...
confirm(a!=x, . . .)	✓	✓	confirm a!=x, ...
confirm(name=none)	×	✓	confirm that no suitable entity is
confirm(a=dontcare,...)	✓	✓	confirm that a is a “don’t care” value
confreq(a=x,...,c=z,d)	✓	×	confirm a=x, ... , c=z and request value of d
select(a=x, a=y)	✓	×	select either a=x or a=y
affirm()	✓	✓	simple yes response
affirm(a=x,b=y, . . .)	✓	✓	affirm and give further info a=x, b=y, ...
negate()	✓	✓	simple no response
negate(a=x)	✓	✓	negate and provide the corrected value for a
negate(a=x,b=y, . . .)	✓	✓	negate(a=x) and give further info b=y, ...
deny(a=x,b=y)	×	✓	inform that a!=x and give further info b=y, ...
repeat()	✓	✓	request to repeat last act
help()	×	✓	request for help
restart()	×	✓	request to restart
null()	✓	✓	null act, does nothing