HAL open science

Reinforcement learning for Dialogue Systems optimization with user adaptation.

Nicolas Carrara

To cite this version:

Nicolas Carrara. Reinforcement learning for Dialogue Systems optimization with user adaptation.. Artificial Intelligence [cs.AI]. Ecole Doctoral Science pour l'Ingénieur Université Lille Nord-de-France, 2019. English. NNT : . tel-02422691

HAL Id: tel-02422691

https://theses.hal.science/tel-02422691v1

Submitted on 23 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université des Sciences et des Technologies de Lille
Ecole Doctorale Sciences Pour L'ingénieur

Thèse de Doctorat
Spécialité: **Informatique**
Présentée par **Nicolas Carrara**

Reinforcement learning for Dialogue Systems optimization with user adaptation

sous la direction du **Pr. Olivier Pietquin**
et l'encadrement du **Dr. Romain Laroche**
ainsi que du **Dr. Tanguy Urvoy**

Soutenue publiquement à Villeneuve d'Ascq, le 18 Décembre 2019 devant le jury composé de :

Pr. Philippe **Preux**	Université de Lille	Président du jury
Dr. Alain **Dutech**	INRIA	Rapporteur
Pr. Fabrice **Lefèvre**	Université d'Avignon	Rapporteur
Dr. Lina Maria **Rojas-Barahona**	Orange Labs	Examinatrice
Pr. Olivier **Pietquin**	Université de Lille, Google Research	Directeur de thèse
Dr. Romain **Laroche**	Microsoft Research	Encadrant industriel
Dr. Tanguy **Urvoy**	Orange Labs	Encadrant industriel

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL),
UMR 9189 Équipe SequeL, 59650, Villeneuve d'Ascq, France

Abstract

The most powerful artificial intelligence systems are now based on learned statistical models. In order to build efficient models, these systems must collect a huge amount of data on their environment. Personal assistants, smart-homes, voice-servers and other dialogue applications are no exceptions to this statement. A specificity of those systems is that they are designed to interact with humans, and as a consequence, their training data has to be collected from interactions with these humans. As the number of interactions with a single person is often too scarce to train a proper model, the usual approach to maximise the amount of data consists in mixing data collected with different users into a single corpus.

However, one limitation of this approach is that, by construction, the trained models are only efficient with an "average" human and do not include any sort of adaptation; this lack of adaptation makes the service unusable for some specific group of persons and leads to a restricted customers base and inclusiveness problems. This thesis proposes solutions to construct Dialogue Systems that are robust to this problem by combining Transfer Learning and Reinforcement Learning. It explores two main ideas:

The first idea of this thesis consists in incorporating adaptation in the very first dialogues with a new user. To that extend, we use the knowledge gathered with previous users. But how to scale such systems with a growing database of user interactions? The first proposed approach involves clustering of Dialogue Systems (tailored for their respective user) based on their behaviours. We demonstrated through handcrafted and real user-models experiments how this method improves the dialogue quality for new and unknown users. The second approach extends the Deep Q-learning algorithm with a continuous transfer process.

The second idea states that before using a dedicated Dialogue System, the first interactions with a user should be handled carefully by a safe Dialogue System common to all users. The underlying approach is divided in two steps. The first step consists in learning a safe strategy through Reinforcement Learning. To that extent, we introduced a budgeted Reinforcement Learning framework for continuous state space and the underlying extensions of classic Reinforcement Learning algorithms. In particular, the safe version of the Fitted-Q algorithm has been validated, in term of safety and efficiency, on a dialogue system tasks and an autonomous driving problem. The second step consists in using those safe strategies when facing new users; this method is an extension of the classic ε-greedy algorithm.

Résumé

Les systèmes d'intelligence artificielle les plus puissants utilisent désormais des modèles statistiques. Afin de construire des modèles efficaces, ces systèmes doivent collecter une quantité substantielle de données issues de l'environnement. Les assistants personnels, maisons connectées, serveurs vocaux et autres systèmes de dialogue ne font pas exception. Ces systèmes ont pour vocation d'interagir avec des humains, et pour cela, leurs données d'apprentissage se doivent d'être collectées avec ces mêmes humains. Parce que le nombre d'interactions avec une seule personne est assez faible, l'approche usuelle pour augmenter le jeu de données consiste à agréger les données de tous les utilisateurs.

Une des limitations de cette approche vient du fait que, par construction, les modèles entraînés ainsi ne sont efficaces qu'avec un humain "moyen" et n'incluent pas de système d'adaptation ; cette faiblesse entraîne la restriction du service à certains groupes de personnes; Par conséquent, cela réduit l'ensemble des utilisateurs et provoque des problèmes d'inclusion. La présente thèse propose des solutions impliquant la construction de systèmes de dialogue combinant l'apprentissage par transfert et l'apprentissage par renforcement. La thèse explore deux pistes de recherche :

La première consiste à inclure un mécanisme d'adaptation dès les premières interactions avec un nouvel utilisateur. Pour ce faire, nous utilisons la connaissance accumulée avec des utilisateurs déjà connus du système. La question sous-jacente est la suivante : comment gérer l'évolution du système suite à une croissance interrompue d'utilisateurs et donc de connaissance? La première approche implique le clustering des systèmes de dialogue (chacun étant spécialisé pour un utilisateur) en fonction de leurs stratégies. Nous démontrons que la méthode améliore la qualité des dialogues en interagissant avec des modèles à base de règles et des modèles d'humains. La seconde approche propose d'inclure un mécanisme d'apprentissage par transfert dans l'exécution d'un algorithme d'apprentissage profond par renforcement, Deep Q-learning.

La seconde piste avance l'idée selon laquelle les premières interactions avec un nouvel utilisateur devraient être gérées par un système de dialogue sécurisé et précautionneux avant d'utiliser un système de dialogue spécialisé. L'approche se divise en deux étapes. La première étape consiste à apprendre une stratégie sécurisée avec de l'apprentissage par renforcement. À cet effet, nous proposons un nouveau framework d'apprentissage par renforcement sous contrainte en états continus ainsi que des algorithmes les solutionnant. En particulier, nous validons, en termes de sécurité et d'efficacité, une extension de Fitted-Q pour les deux applications sous contraintes : les systèmes de dialogue et la conduite autonome. La deuxième étape implique l'utilisation de ces stratégies sécurisées lors des premières interactions avec un nouvel utilisateur ; cette méthode est une extension de l'algorithme classique d'exploration, ε-greedy.

Acknowledgement and Thanks

Je tiens tout d'abord à remercier Olivier pour m'avoir aidé à faire émerger des idées nouvelles et à leur donner vie ensemble dans une ambiance de grande camaraderie. Un merci à Romain pour m'avoir guidé dès le début de la thèse ; sa rigueur et ses précieuses relectures ont grandement contribué à la qualité de mes travaux. J'aimerais aussi remercier Tanguy pour son enthousiasme et sa volonté d'approndir les concepts les plus techniques, et Jean-Léon pour son approche alternative et pédagogique de mes travaux.

Je voudrais ensuite remercier Alain Dutech et Fabrice Lefèvre d'avoir accepté d'être les rapporteurs de cette thèse. Mes remerciements s'adressent aussi à Lina Maria Rojas-Barahona et Philippe Preux, respectivement examinatrice et Président du jury.

Je tiens également à remercier mes collègues, notamment Edouard pour sa contribution enrichissante à notre dernier projet, Merwan, Ronan, Hatim, Mathieu, Florian, Xuedong, Guillaume, Julien, Omar, Pierre, et Odalric pour les discussions techniques qui m'ont permis d'avancer. Je souhaite aussi remercier les personnes avec qui j'ai partagé mon quotidien, notamment Masha, William, Tatiana, Lilian, Michal, Mateo, Emilie, Amélie, et Phillipe.

En outre, je n'oublie pas mes collègues d'Orange Labs, en particulier Myriam pour la dimension humaine de son encadrement.

Enfin, je suis reconnaissant envers mes proches pour leur soutien et leurs encouragements tout au long de mon cursus universitaire.

Contents

1 Symbols 13

Acronyms 15

2 Introduction 17

2.1 History of dialogue systems 18

2.2 A challenge for modern applications 21

2.3 Contributions 22

2.4 Publications 23

2.5 Outline 24

I Task-oriented Dialogue Systems

3 The pipeline 27

3.1 A modular architecture 27

3.2 On the slot-filling problem 29

3.2.1 Settings 29

3.3 The Dialogue Manager 30

4 Training the Dialogue Manager with RL 33

4.1 Assuming a given dialogue corpus 39

4.2 Online interactions with the user 40

4.3 To go beyond 43

5 User adaptation and Transfer Learning 45
5.1 The problem of Transfer Learning 45
5.2 State-of-the-art of Transfer Learning for Dialogue Systems 47
5.2.1 Cross domain adaptation 48
5.2.2 User adaptation 49
5.2.3 An aside on dialogue evaluation 50

II Scaling up Transfer Learning

6 A complete pipeline for user adaptation 55
6.1 Motivation 55
6.2 Adaptation process 56
6.2.1 The *knowledge-transfer* phase 56
6.2.2 The *learning* phase 57
6.3 Source representatives 57
6.4 Experiments 58
6.4.1 Users design 60
6.4.2 Systems design 61
6.4.3 Cross comparisons 61
6.4.4 Adaptation results 63
6.5 Related work 65
6.6 Conclusion 66
6.7 Discussion 66

III Safe Transfer Learning

7 The Dialogue Manager as a safe policy 69
7.1 Motivation 70
7.1.1 A remark on deterministic policies 71
7.2 Budgeted Dialogue Policies 71
7.3 Budgeted Reinforcement Learning 75
7.3.1 Budgeted Fitted-Q 75
7.3.2 Risk-sensitive exploration 75
7.4 A scalable Implementation 76
7.4.1 How to compute the greedy policy? 76
7.4.2 Function approximation 77
7.4.3 Parallel computing 78

7.5 Experiments **78**
7.5.1 Environments . . . 80
7.5.2 Results . . . 82
7.5.3 Budgeted Fitted-Q policy executions . . . 82
7.6 Discussion **86**
7.7 Conclusion **86**

8 Transfering safe policies . . . 89
8.1 Motivation **89**
8.2 ε-safe **89**
8.3 Experiment **90**

IV Closing

9 Conclusion . . . 95
9.1 Contributions **95**
9.2 On the long run **96**

10 Appendices . . . 99
.1 Proofs of Main Results **99**
.1.1 Proposition 7.2.1 . . . 99
.1.2 Theorem 7.2.2 . . . 100
.1.3 Proposition 7.2.3 . . . 101
.1.4 Theorem 7.2.4 . . . 101
.1.5 Theorem 7.2.5 . . . 102
.1.6 Proposition 7.4.1 . . . 106
.2 Parameters **108**
.3 Repoducibility **110**
.3.1 Instructions for reproducibility . . . 110
.3.2 The Machine Learning reproducibility checklist . . . 110

A Continuous transfer in Deep Q-learning . . . 113
A.0.1 Deep Q-learning . . . 113
A.1 The transfer phase. **114**
A.1.1 Auto-Encoders . . . 114
A.1.2 Using the Temporal Difference error . . . 114
A.2 The learning phase **114**
A.2.1 Transferring the transition . . . 114
A.2.2 Transferring the network . . . 115
A.3 Conclusion **116**
A.3.1 Discussion . . . 116

Bibliography 117

Index 137

List of items

Figures

2.1 The Voder demonstration 19

3.1 The architecture of a modular Spoken Dialogue System 28
3.2 The pipeline view from the Dialogue Policy 31

4.1 Example of Markov Decision Processes 35
4.2 The Dialogue Manager cast as a Reinforcement Learning problem. . . . 39

5.1 Objectives of Transfert Learning (Langley 2006; Lazaric 2012) 47

6.1 Adaptation process 56
6.2 Some projections of policies optimised versus human-model users 62
6.3 Dialogue quality in the handcrafted and human setup 64

7.1 Example of relaxed Budgeted Markov Decision Process 71
7.2 Representation of $\overline{\pi}_{\text{hull}}$. 77
7.3 A Budgeted Fitted-Q Neural-Network 78
7.4 Results on `Corridors` 83
7.5 Results on `slot-filling` and `highway-env` 84
7.6 Calibration for Lagragian Relaxation. 87

8.1 ε-safe algorithm. 90
8.2 Half-Gaussian distribution of p values. 90
8.3 Performance of the greedy policies. 91

1 Concavity Example 102

2 We represent the range of possible solutions ${Q_r}^{2,*}$ for any $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1)$, given $\overline{Q}_1 \in \mathcal{L}_\lambda$ 104
3 We represent a section $[\beta - \kappa, \beta + \kappa]$ of $\mathcal{F}^1$ and $\text{Ball}(\mathcal{F}^1, R)$. We want to bound the range of ${Q_r}^{2*}$. 105

A.1 Counter example for the Temporal Difference error solution 115

Tables

5.1 Transfer Learning for Dialogue Systems 51

6.1 Actions distributions of humans 60
6.2 Cross comparison between handcrafted users and systems 63
6.3 Cross comparison between human-model users and systems 63

7.1 Parameters of `Corridors` 80
7.2 Parameters of `Slot-Filling` 81
7.3 Parameters of `highway-env` 82
7.4 Example of Dialogue Policies executions 85

1 Algorithms parameters for `Corridors` 108
2 Algorithms parameters for `slot-filling` 109
3 Algorithms parameters for `Highway-Env` 109

Algorithms

1 Value-Iteration 37
2 Policy-Iteration 38

3 Budgeted Value-Iteration 74
4 Budgeted Fitted-Q 75
5 Risk-sensitive exploration 76
6 Convex hull policy $\overline{\pi}_{\text{hull}}(\overline{a}|\overline{s}; \overline{Q})$ 77
7 Scalable Budgeted Fitted-Q 79

1. Symbols

$\mu_\perp$ Gaussian mean for the computing of the speech recognition score when the utterance is missunderstood. 58–61, 63–65, 81
C Constraint function. 70, 72, 74, 81, 86, 101, 106
G_c Discounted return for the constraint signal. 70–73, 80, 82
G Discounted return for the reward signal. 36, 59, 70–73, 80, 82
K Number maximum of iterations. 40, 61, 65
N Number of transitions. 34, 39, 40, 46, 57, 58, 61, 63, 108, 109
P Transition function. 33, 34, 36–40, 45, 72, 74, 86, 101, 114, 116
Q_c Constraint Q-function. 72–74, 77–79, 82, 101–106
Q_r Reward Q-function. 12, 72–74, 77–79, 100–107
Q Q-function (action-value function). 36–38, 40–42, 46, 48, 50, 51, 71, 74, 89, 96, 113–115
R Reward function. 33, 34, 36–40, 45, 70, 72, 74, 81, 86, 100, 101, 106, 116
V_c Constraint value function. 72, 73, 101
V_r Reward value function. 72, 73, 100
V Value function. 65
α Learning rate. 41, 42
β' Next budget. 72
β_a Budget allocated to an action. 71, 72, 75, 76, 78, 102
β Budget. 12, 70–73, 75–79, 82, 85–87, 101, 103–108
δ Dirac function. 72, 77, 106
ε Probability of taking the random action in the ε-greedy exploration procedure. 11, 13, 23, 24, 41, 42, 57, 61, 65, 75, 76, 78, 89, 90, 92, 96, 137
γ Discount factor. 34, 36–38, 40–42, 59–61, 65, 70, 72–75, 78, 79, 100–105, 108, 109
κ An extra small value (to avoid overriding epsilon symbol). 12, 101, 102, 104, 105
$\mathcal{A}$ Set of actions. 34, 36–38, 40–42, 45–47, 70, 71, 74–79, 101, 106
$\mathcal{B}$ Binomial distribution. 58, 76

$\mathcal{D}$ Batch of data. 39, 40, 75, 76, 78–80, 108, 109, 114
$\mathcal{K}$ Space of knowledge. 46
$\mathcal{M}$ Measure. 34, 36, 70, 72, 73, 100, 106
$\mathcal{N}$ Normal distribution. 58, 81
$\mathcal{S}$ Set of states. 34, 36–38, 40, 45–47, 70, 71, 74, 75, 78, 86, 101
$\mathcal{T}$ Bellman operator ($*$ for optimality, π for evaluation). 36–38, 40
$\mathcal{U}$ Uniform distribution. 76
$\mathscr{B}$ Set of admissible budgets. 70, 71, 75, 76, 78, 79, 108, 109
$\mu_\top$ Gaussian mean for the computing of the speech recognition score when the utterance is understood. 58–61, 63–65, 81
ν Speech Recognition Score. 58, 59, 61, 63, 81, 85, 108
$\overline{G}$ Augmented return. 72, 99
$\overline{P}$ Augmented transition function. 72, 73, 75, 100, 102, 105
$\overline{Q}$ Augmented Q-function. 12, 72–79, 82, 99–107
$\overline{R}$ Augemented Reward function. 72, 100
$\overline{R}$ Augmented reward. 72, 73
$\overline{V}$ Augmented value function. 72–74, 99–101
$\overline{\Pi}$ Set of budgeted policies. 72–74, 100, 101, 106, 107
$\overline{\mathcal{A}}$ Set of augmented actions. 71–73, 75, 77, 79, 99–106
$\overline{\mathcal{S}}$ Set of augmented states. 71–75, 79, 100–106
$\overline{\mathcal{T}}$ Augmented Bellman operator ($*$ for optimality, π for evaluation). 72–76, 78, 82, 100–103, 105, 106
$\overline{\pi}^*$ Optimal budgeted policy. 75
$\overline{\pi}$ Budgeted policy. 11, 12, 72–78, 82, 99–102, 105, 106
$\overline{\Xi}$ Augmented projection operator. 75
ϕ Features function. 40, 42, 61
π^* Optimal policy. 36–38, 65, 70, 71
π Policy. 36–38, 41, 46, 57, 58, 65, 70, 71, 74, 80, 82, 86, 114
$\mathscr{U}$ Set of users. 46, 58, 113, 114
θ Parameters to learn. 40, 42, 75, 82
$\top$ Matrix transposition. 40, 42
υ Stopping criterion for Value Iteration like algorithm, when successive parameters are close to each other. 37, 40, 61, 65
ξ Speech Error Rate. 58, 60, 61, 64, 65, 81
i Index of a transition. 33, 34, 36, 39–42, 46, 57, 63, 75, 79, 114
k State of an iterative process. 37, 38, 40, 42
Ξ Projection operator. 38, 40

Acronyms

kNN k Nearest Neighbours. 60, 61

A.I. Artificial Intelligence. 17, 97
API Approximate Policy Iteration. 39
ASR Automatic Speech Recognition. 18, 19, 21, 27, 49, 58, 90
AVI Approximate Value Iteration. 38

BFTQ Budgeted Fitted-Q. 23, 24, 71, 75, 76, 78, 80, 82, 96, 108, 109
BMDP Budgeted Markov Decision Process. 24, 69–72, 74, 75, 78, 80, 86, 101

CMDP Constrainted Markov Decision Process. 70, 71, 80, 86, 106
CPU Central Processing Unit. 23, 76, 79, 110, 111

DM Dialogue Manager. 18, 19, 21, 24, 27–30, 33, 39–41, 43, 45, 46, 48, 49, 95
DP Dialogue Policy. 28–31, 33, 36, 41, 42, 46, 48, 49, 59, 86, 96, 114
DQN Deep Q-learning. 2, 3, 23, 43, 66, 96, 113–116
DRL Deep Reinforcement Learning. 23, 43, 69, 71, 86
DS Dialogue System. 2, 17–24, 29, 30, 33, 39, 47, 49, 65, 66, 69, 87, 89, 90, 95–97
DST Dialogue State Tracking. 21, 28–30, 33, 48, 50, 51, 61
DSTC Dialogue State Tracking Challenge. 21, 29, 39, 96

FTQ Fitted-Q. 2, 3, 40, 42, 50, 57, 58, 61, 64, 65, 75, 80, 82, 84, 86, 108, 109
FVI Fitted-Value-Iteration. 38, 40, 75

GAN Generative Adversarial Network. 19
GD Gradient Descent. 42
GP Gaussian Process. 48–50
GPU Graphics Processing Unit. 17, 110, 111

HMM Hidden Markov Model. 19

iid independent and identically distributed. 113

LS Least Squares. 40, 42
LSPI Least Squares Policy Iteration. 40, 51
LSTM Long Short-Term Memory. 19, 27–29

MAB Multi-Armed Bandit. 55, 57
MDP Markov Decision Process. 21, 31, 33, 34, 36, 45, 47, 55, 57, 65, 69–71, 114
ML Machine Learning. 17, 18, 20, 49, 99

NDG Negociation Dialogue Game. 23, 50, 58, 59, 66, 96
NLG Natural Language Generator. 19, 27–29
NLU Natural Language Understanding. 19–21, 27, 28, 30, 81
NN Neural Network. 19, 20, 27, 28, 39, 40, 46, 48, 71, 77, 78, 82, 108, 109, 113

PG Policy Gradient. 43
PI Policy Iteration. 37–40
POMDP Partially Observable Markov Decision Process. 21, 29

RBM Restricted Boltzmann Machine. 114
RL Reinforcement Learning. 2, 17, 22–24, 30, 31, 36, 38–43, 45, 46, 55, 69–71, 74, 77, 80, 86, 89, 95–97, 113, 116
RNN Recurrent Neural Networks. 19, 21, 27, 28, 30

SARSA State–Action–Reward–State–Action. 50
SDS Spoken Dialogue System. 18, 19, 21, 22, 27, 29, 30, 50, 71
SER Sentence Error Rate. 58, 59, 64, 81
SGD Stochastic Gradient Descent. 42
SL Supervised Learning. 46, 115
SRS Speech Recognition Score. 27, 30, 34, 36, 38, 58, 59, 61, 63, 69, 81, 82, 86, 108

TD Temporal Difference. 41, 113–116
TL Transfer Learning. 2, 18, 22–24, 41, 42, 45–49, 55, 95, 96, 113
TTS Text To Speech. 18, 19, 27, 29

UCB Upper Bound Confidence. 50, 57

VI Value Iteration. 37, 38

2. Introduction

In the past few years, Artificial Intelligence (A.I.) has experienced an unprecedented growth in both industry and research. This was made possible by advances in Machine Learning (ML) and statistics but also thanks to the computing capacities of modern Graphics Processing Unit (GPU). A broad spectrum of domains enjoys the fallouts: medicine, law, digital marketing, finance, automobile industry, and so on. The overwhelming majority of applications involves predicting outcome given an input in a stateless fashion. For example, using image recognition, a radiologist may get help finding tumours on an x-ray image; given a sentence, an automatic translation tool infers the same sentence in an other language; a trading black-box predicts the next variation of a stock with the knowledge of all the past variations in stock exchange. When the task involves interacting with the environment in a sequential fashion, the tools used to solve all the aforementioned problems do not suffice. This class of problems includes - among others - autonomous driving, power-grid management, online advertising, video games, robotics, and the domain this work focuses on: dialogue.

In dialogue applications, two agents interact with each others. A classic use-case is an agent, called Dialogue System (DS), interacting with a human user. To that end, one must designs the strategy the DS will adopt. While pioneer methods involve rule-based DSs, recent solutions involve learning the DS strategy, or *policy*, using the Reinforcement Learning (RL) framework. The basic idea is that an *agent* (the DS) interacts with the *environment* (the user). As it is interacting, the agent *reinforces* its knowledge of the environment and adapts its policy accordingly. One drawback of this method is it may need a substantial amount of interactions to construct a good enough policy. This statement is especially true during early interactions. Correspondingly, if a human tries to solve a completely new task, for instance, operating weightlessness with a full spacesuit, he will fail or, at the very best, face some issues in his first interactions, then eventually adapt to the new environment. Because space missions are expensive and time is critical, and for probably a lot of additional factors out of the scope of this thesis, an astronaut cannot

afford to learn from scratch this new task. To remedy this situation, astronauts are trained to execute a set of tasks equipped with their suit, in a weightless pool on Earth. That way, they will be able to *transfer* their knowledge gathered in the pool environment to quickly adapt in the space environment. Now, we consider that the astronaut represents the DS, the pool environment represents a young person used to new technologies and the space environment represents an elderly person with a very weak control over new technologies. If one trains a DS enough to interact flawlessly with the young person, then he may be able to transfer the data, the model or even the learnt world representation (*features*) to a new DS, in order to speed-up the learning when interacting with the elderly person. In ML, this process is called Transfer Learning (TL) and this thesis makes the hypothesis that TL should be a good fit to improve the learning of DSs.

2.1 History of dialogue systems

In the collective imagination, a robot should walk, execute tasks, interact, and seamlessly talk with humans. Each of these abilities is actually a whole domain of research. For now, the separation is clear between each domain[1]. Science fiction introduced intelligent agents that can dialogue; while Hal9000 (Clarke 1968) or Kit (Larson 1986) talk and control a fully mechanical body (a spaceship and a car respectively), the sole purpose of the Multivac (Asimov 1961) is question answering. In the same fashion, this thesis focuses on building intelligent agents, known as DSs, where the sole purpose is to interact with the human through the voice and using text. To build a DS, the designer must solve the following challenges: process the human voice, extract meanings, decide what to respond, create a proper semantic for the response and, finally, generate the artificial speech signal for the response. While this thesis focuses on the decision part (also called Dialogue Manager (DM)), a lot of research has been conducted on the other domains, independently from DS in a whole.

Processing and generating speech

While text-based DSs process text information and produce textual answers, Spoken Dialogue System (SDS) interact with the users through the voice channel. The challenging task for SDSs is then to understand and generate spoken sentences (text to audio and audio to text). Solutions incrementally more and more impressive trough the years have been proposed from the signal processing community in the 20th century. For these particular tasks, two modules are involved and detailed in Chapter 3, the Automatic Speech Recognition (ASR) module used to process user spoken language, and the Text To Speech (TTS) module to produce the system spoken response.

The first solution to the speech synthesis problem, known as TTS, was the Voice Operating Demonstrator known as Voder (Homer et al. 1939). It is an electronic machine able to synthesise human speech using a dedicated keyboard. It is based on Dudley's work at Bell Lab on the vocoder, a tool to analyse and synthesise speech. The Voder has been introduced at the 1939 New York World's Fair as show in Figure 2.1. In 1950, Dr. Franklin S. Cooper from Haskins Laboratories released the final version of the Pattern playback (Rubin et al. 2019). This device generates sounds given spectrograms of speech

[1] Even if one could argue that everything is connected, as for example, it may be hard to converse with a robot that is running around.

patterns. In the end of the 20th century, the dominating speech synthesizer was DECtalk. The system is based on early work on KlattTalk (Dennis 1987). The system itself was a standalone pluggable on any telephone facilitating vocal servers. Recently, DeepMind showed impressive results with WaveNet (Van Den Oord et al. 2016), a deep Neural Network (NN) for generating raw audio waveforms. The network is able to generate speech in many languages but also novel music fragments. The same company introduced very recently a Generative Adversarial Network (GAN) approach for TTS (Bińkowski et al. 2019).

The earliest attempt at designing an ASR module was successfully achieved by the Elmwood Button Company with Radio Rex. It was a toy dog built to react to simple sound patterns, like "Rex". In the late 1940s, the military got involved, when the U.S. Department of Defense sponsored the first researches in speech recognition in order to process automatically intercepted Russian messages. Following up, numerous systems were able to processing digits (Davis et al. 1952), vowels (W. Forgie et al. 1959) and finally words (Ben 1966). It is in the 1970's that the Defense Advanced Research Projects Agency (from the U.S. Department of Defense) ran a massive speech recognition program involving well-established research groups, this attempt was unsuccessful but it led to a new area of research around Hidden Markov Models (HMMs) (Jelinek 1976). HMMs were a first step in the art of learning statistical models based on data. HMMs were a reference in ASR until the perceptron (Rosenblatt 1958) expanded into NNs. HMM approaches involve different modules to handle pronunciation, acoustics and language model. NNs shift the problem to an end to end approach, i.e. sound signal to sentence. To that extent, Recurrent Neural Networks (RNN) (Rumelhart et al. 1986), and later Long Short-Term Memory (LSTM) (Hochreiter et al. 1997), have been applied to predict sequences of words (Graves et al. 2014).

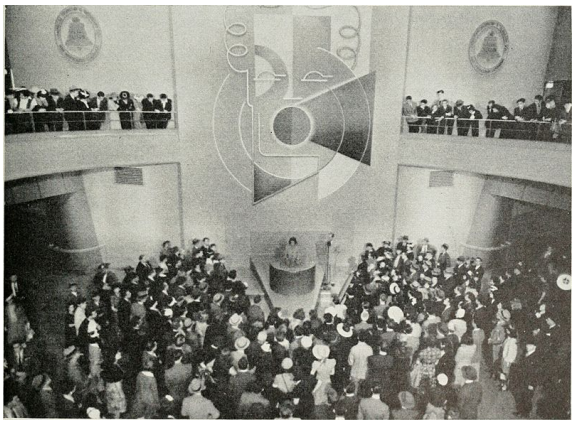


Figure 2.1: The Voder demonstration

On the semantic problematic

Two components of a DS, the Natural Language Generator (NLG) and the Natural Language Understanding (NLU) handle most of the semantics involved in the dialogue process. The NLU transforms text to concepts/semantics, the NLG does the opposite.

Research in NLGs started growing in 1970. Several applications benefited from the advances as text summary (Goldberg et al. 1994) and documentation generation (Lavoie et al. 1977). The SDSs have been using template-based NLG for a long time. The first fully automated NLG for SDSs were developed in the early 2000s (Oh et al. 2000; Rambow et al. 2001). In Lemon (2011), the NLG module from the SDS and the DM (as discussed in Chapter 3) are jointly optimised. Later, Perera et al. (2017) survey described 12 years of advances in NLGs research. More recently, NNs have been used to automatically generate behavior explanations of an autonomous agent (Ehsan et al. 2018).

Meanwhile, in the same period, research around NLU has been conducted. In 1954, the Georgetown–IBM experiment has been conducted in order to process automatic translations of Russian sentences to English using grammar rules. Then, during his Ph.D thesis, Daniel Bobrow developped STUDENT, a program solving algebra word problems (Bobrow 1964). In 1969, Roger Schank introduced the conceptual dependency theory (Schank et al. 1969): the idea was to share the same representation for two sentences with equivalent meaning. Later, ML methods have been used for NLU with, for example, statistical pattern-matching (Allen 1995). The next millennium saw IBM Watson, supercomputer champion of the Jeopardy?!, using the deepQA technology (Ferrucci et al. 2010). Finally, more recent approaches use NNs.

The Turing-Test

The ground basis of DSs, and computer science in general, has been established by Alan Turing. He was a mathematician and logician and he is notably known as the first computer scientist in the modern term's meaning. His well known contributions are the Turing-Machine (Turing 1936) and the Turing-Test (Turing 1950) both in scientific world and popular culture[2].

Quote 1 gives a rough idea of what is the Turing-Test. This test is inspired by the imitation game, where an interrogator must determinate the gender of a human, based only on written interrogations. While the imitation game discriminates males from females, the Turing-Test discriminates humans from machines. This test is designed to measure the human-likelihood of a DS. A human, the tester, dialogues independently with two agents hidden in a box. One agent is another human, the other is the DS. If the tester is unable to tell which is the human and which is the artificial system, then the test is successful.

> **Bliss:** Isn't it possible I may be so cleverly artificial that in every respect, from largest to smallest, I am indistinguishable from the natural. If I were, how could you tell the difference between me and a true human being? ? [...]
>
> **Janov:** It seems to me, then, that a robot that can in no way be distinguished from a human being is a human being. If you were such a robot, you would be nothing but a human being to me.

Quote 1: A dialogue between Bliss and Janov Perolat about discriminating Bliss as a robot or not, in Foundation and Earth (Asimov 1986)

Recent attemps in the dialogue community proposed automatic almost-Turing-Test or scores. The BLEU score (Papineni et al. 2002) was introduced to measure the human-likeness of an automatic translation. The ROUGE score (Lin 2004) evaluates the quality of an automatic text summary. ADEM (Lowe, Noseworthy, et al. 2017) is an automatic dialogue evaluator learnt on datasets of human evaluations.

[2] To name a few: in the movie Blade Runner (Scott 1982), the Voight-Kampff test is a Turing-like test to identify replicants (robots). More recently, the whole plot of Ex-machina (Isaac 2015) is around the Turing-Test. In the Imitation Game (Moore 2014), we learn how Alan Turing and his team deciphered the Enigma German machine using a dedicated electro-mechanical device (called the Turing-bombe), potentially shortening World War II of a few years.

Dialogue Systems

Previously covered technologies were not DSs strictly speaking but rather individual technologies later used as components of DSs. The first actual textual DS which was capable of attempting the Turing-Test is ELIZA (Weizenbaum 1966). It is a text-based DS (chatbot) used to mimic a Rogerian psychotherapist (Rogers 1942); it reformulates vaguely what the patient said in order to express empathy. In 1972, the psychiatrist Kenneth Colby created PARRY (Colby 1975), a textual DS simulating a person with paranoid schizophrenia. This program also passed the Turing-Test when psychiatrists were unable to discriminate PARRY from a human patient better than random guessing. In 1972, three major technologies of the 70s met; during the International Conference on Computer Communication, PARRY and ELIZA exchanged through the ARPANET, the ancestor of the Internet. We can already distinguish two types of DSs, a chatbot, ELIZA, and an user-model, PARRY, which simulates a real human user. In 1977, Daniel Bobrow introduced the Genial Understander System as one of the first task-oriented DS in the form of a travel agent (Bobrow et al. 1977). In 1995, the Massachusset Institute of Technology released Voyager (Glass et al. 1995), an SDS for urban navigation. It was quickly followed by Jupiter, from the same laboratory, a telephonic SDS for weather information (Zue et al. 2000). Meanwhile, Orange created ARTIMIS (Sadek et al. 1997), a DS based on the idea of rational agency, where the dialogue can be seen as rational interactions using logical axioms (Cummings 2010). In 1996, the company Charles Schwab released the eSchwab, an internet service for stock information (Cortada 2005). They designed the service as a DS using all the aforementioned modules (ASR, NLU etc) (Pietquin 2004).

Marilyn Walker was the first to cast the DS's strategy as a stochastic optimization problem (Walker 1993), followed closely by Biermann et al. (1996), and later by Walker et al. (1998), with real users experiments. Meanwhile, Esther Levin was the first to formally describe dialogue management as a Markov Decision Process (MDP) problem (Levin and Pieraccini 1997). Then Partially Observable Markov Decision Processes (POMDPs) (Karl Johan 1965) have been extensively used for dialogue management (Roy et al. 2000; S. J. Young et al. 2013). The DM uses belief tracking (or Dialogue State Tracking (DST)), i.e. summary of the current state of the dialogue using dialogue history. It led to the Dialogue State Tracking Challenge (DSTC) (Jason D. Williams et al. 2013), a competition evaluating the best models for belief tracking. RNNs did very well at this task (Henderson et al. 2013).

2.2 A challenge for modern applications

Dialogue Systems are autonomous agents intended to converse with humans (or even another DS). The state-of-the-art algorithms are now data-driven solutions (Lemon and Pietquin 2012); this ability to automatically process voice or text from a user can improve considerably any application based on a human-machine interface. In the movie Her (Jonze 2013), the dialoguing entity, an overpowered vocal assistant pretty close to Artificial General Intelligence, can process or deliver any kind of information and comes along with a lot a features that cover anything a human could offer and more. In real life, for now, dialogue applications are divided in three parts, corresponding to three kinds of problems (Gao et al. 2019):

Social chat

The DSs converses with the user like it would do passing the Turing-Test. The conversation is mainly chit-chat with some recommendations and there is no well-defined goal. The purpose of these systems is entertainment-driven. To that end, systems are mainly trained to reproduce human conversation using predictive models or retrieval-based methods (H. Chen et al. 2017). Mitsuku (Worswick 2005) and Rose (Bruce 2011) are examples of chit-chat applications available on the market. Chit-chat can be also involved in the other dialogue problems; in the online shopping domain, while the DS actually performs a task, 80% of the utterances are chit-chat messages (Z. Yan et al. 2017).

Question answering

In a more pragmatic way, the DS must provide a direct and concise answer to user queries containing all the information needed. Most of the research focuses solely on query question answering. So the system extracts keywords from the user query with semantic parsing then consults a database (web documents or knowledge bases such as sales and marketing datasets) to get the information. It reformulates this information with natural language paraphrasing in a human way. State-of-the-art vocal assistants (Google assistant, Amazon Alexa, etc) solve this kind of problems (Neustein et al. 2013). Recently, OpenAI team introduced GPT-2 (Radford et al. 2019), a Transformer architecture (Vaswani et al. 2017) that is able to do extremly realistic question answering and text-autocompletion.

Task completion

The DS must achieve a task based on the information provided by the user. It usually takes multiple dialogue turns to gather all the information needed thus involving sequential decision making paradigm (Chapter 4). The user tasks, called domains, may take various forms. A DS can collect medical information about the user then pass it to his doctor (Terry 2019). In a similar way, it can collect information about a user's business then generate a suitable website (Saeed 2019). It could also turn the light on or set the schedule of the washing machine in a smart home. In the car, using a DS to setup the GPS trajectory lets the user free to keep an eye on the road. More recently, Cambridge Dialogue System Group released PyDial, a full framework for SDS (Ultes, Rojas-Barahona, et al. 2017) for restaurant reservation (among other domains). Following those ideas, this thesis focuses on solving task-driven problems using task-oriented DSs. The architecture of these systems is thoroughly detailed in Chapter 3.

2.3 Contributions

As explained at the very beginning of this manuscript, we focus our attention primarily on how to jumpstart the performances of a freshly created DS confronted with an unknown user on a task-completion problem. This problematic can be solved via TL and this thesis explores two basic paradigms. Both cast the dialogue as an RL problem but they differ with respect to the safety.

In the first approach, we do not take the safety into account. We propose two different directions to try and solve the problem: the first proposed method focuses on finding from a pool of previously learned DSs, which DS is the more efficient with a new user. The efficiency has no notion of safety and is guided by a signal only capturing the completion of the task. Then, we transfer the knowledge of this source DS as we call it, to the target

DS (the freshly created DS that will be specialised in dialoguing with the new user). This idea has been introduced by Genevay et al. (2016) and extended by Carrara, Laroche, and Pietquin (2017) for larger DSs pools. The entire framework has been tested on the Negociation Dialogue Game (NDG) (Laroche and Genevay 2017) on both handcrafted and human-model users; the second method focuses on incorporating a TL into the Deep Q-learning (DQN) algorithm. We believe that transferring continuously the knowledge of the pool of previously learned DSs may greatly improve the learning of online Deep Reinforcement Learning (DRL) policies.

The second approach turns its attention to the safety of the dialogue. Here, safety is defined as the ability to keep the mean frequency of user hanging up under a certain level. Strategies designed with safety in mind naturally enhance the quality of the dialogue and on a meta point of view, retain the user as a regular customer with ease. Indeed, one does not want to see a user stop using the service after a series of bad dialogues. At first, one needs to find a way to design those safe policies. In Carrara et al. (2018a) and Carrara et al. (2018b) we introduce Budgeted Fitted-Q (BFTQ), an RL algorithm designed to construct a parametric dialogue strategy (given a certain hangup frequency) using a dialogue corpus. We introduce the algorithm and give a proof of concept on a simple 2D navigation problem. We scale this algorithm to larger problems using neural network and Central Processing Unit (CPU) parallelism and validate it on a dialogue problem and an autonomous driving problem (Carrara, Leurent, Laroche, Urvoy, Maillard, et al. 2019)[3]. To operate TL, BFTQ itself is not enough because it has no mechanism for user adaptation. To overcome this issue, in Carrara et al. (2018c), we introduce ε-safe, a TL algorithm that uses a previously learnt safe strategy as the knowledge to transfer when designing from scratch a DS for an unknown user.

2.4 Publications

The contributions discussed in the previous paragraph have led to various publications in international conferences. Carrara, Laroche, and Pietquin (2017) has been shared through an oral presentation at the session on negotiation dialogue at the joint SIGdial/SemDial conference. Carrara et al. (2018a) led to an oral presentation and a poster in a workshop in the conference on Uncertainty in Artificial Intelligence. The same work (Carrara et al. 2018b) has been shared at the European Workshop in Reinforcement Learning. It is during the International Conference on Statistical Language and Speech Processing that Carrara et al. (2018c) has been published through a poster format. Finally, Carrara, Leurent, Laroche, Urvoy, Maillard, et al. (2019) has been published at the conference on Neural Information Processing Systems main track.

Carrara, Nicolas, Romain Laroche, Jean-Léon Bouraoui, Tanguy Urvoy, and Olivier Pietquin (2018c). "Safe transfer learning for dialogue applications". In: *International Conference on Statistical Language and Speech Processing* (cited on pages 23, 50, 51).

Carrara, Nicolas, Romain Laroche, and Olivier Pietquin (2017). "Online learning and transfer for user adaptation in dialogue systems". In: *Joint special session on negotiation dialog, Workshop on the Semantics and Pragmatics of Dialogue- Conference of the Special Interest Group on Discourse and Dialogue* (cited on pages 23, 50, 51).

[3] All 3 publications are grouped as one: Carrara, Leurent, Laroche, Urvoy, Bouraoui, et al. (2019)

Carrara, Nicolas, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Jean-Léon Bouraoui, Odalric Maillard, and Olivier Pietquin (2019). "Budgeted Reinforcement Learning in Continuous State Space". In: *Workshop on Safety Risk and Uncertainty in Reinforcement Learning at Conference on Uncertainty in Artificial Intelligence (2018), European Workshop on Reinforcement Learning (2018), and Conference on Neural Information Processing Systems (2019)* (cited on page 23).

2.5 Outline

This manuscript is organized as follow: Part I details the state of the art for task-oriented DSs. Inside Part I, Chapter 3 describes the architecture of such DS, involving the modules already discussed in introduction. Then Chapter 4 explains why the DM is a sequential decision making problem and how it can be cast as an RL problem. And finally, Chapter 5 exhibits several TL techniques that can enhance the DM performances in all the stages of the learning.

Part II lists the several contributions of the thesis to scale DSs to a growing base of users: Chapter 6 introduces a novel method for scaling TL in user adaptation applications.This method extends an existing framework with clustering of DSs.

Then, Part III lists contributions to handle user adaptation in a safe way: Chapter 7 proposes BFTQ, an algorithm to solve Budgeted Markov Decision Process (BMDP) in continuous space. It also adds insights for scaling up the algorithm and tests it on a slot-filling task; Chapter 8 presents the last contribution, ε-safe.

Finally, the conclusion in Chapter 9, followed by the the appendixes in Chapter 10, and an early work in progress in Appendix A brings down the curtain on this thesis.

Task-oriented Dialogue Systems

3 The pipeline 27

3.1 A modular architecture
3.2 On the slot-filling problem
3.3 The Dialogue Manager

4 Training the Dialogue Manager with RL 33

4.1 Assuming a given dialogue corpus
4.2 Online interactions with the user
4.3 To go beyond

5 User adaptation and Transfer Learning 45

5.1 The problem of Transfer Learning
5.2 State-of-the-art of Transfer Learning for Dialogue Systems

3. The pipeline

In the introduction, we saw that during the last century, different but related speech technologies have been developed independently from each other. The major areas of research were about Automatic Speech Recognition (ASR) and Text To Speech (TTS). Following up shortly, Natural Language Generator (NLG) and Natural Language Understanding (NLU) gradually gained interest. Finally, with the addition of the Dialogue Manager (DM), all those technologies have been gathered to construct a modular Spoken Dialogue System (SDS) (Jurafsky et al. 2000). In this chapter, we describe briefly all the involved modules before focusing our attention on the DM.

3.1 A modular architecture

Figure 3.1 described the workflow leading a SDS, processing a speech signal as an input and outputs an information of the same type.

Automatic Speech Recognition

The ASR module processes the speech signal. It produces interpretation hypotheses, as the most probable sentences the user could have said. Each hypothesis is associated with a score we call the Speech Recognition Score (SRS). As recalled in introduction, the state of the art of ASR includes Neural Network solutions as for example the Connectionist Temporal Classification (S. Kim et al. 2018), RNN (Graves et al. 2014) and LSTM (J. Kim et al. 2017). They can also be combined with language modelling (Chorowski et al. 2017; Lee et al. 2018).

Natural Language Understanding

Given those hypotheses, the NLU can extract meanings, and builds a semantic frame corresponding to the last user utterance, sometimes paired with a confidence score. Classic approaches parse the user utterance into predefined handcrafted semantic slots (H. Chen

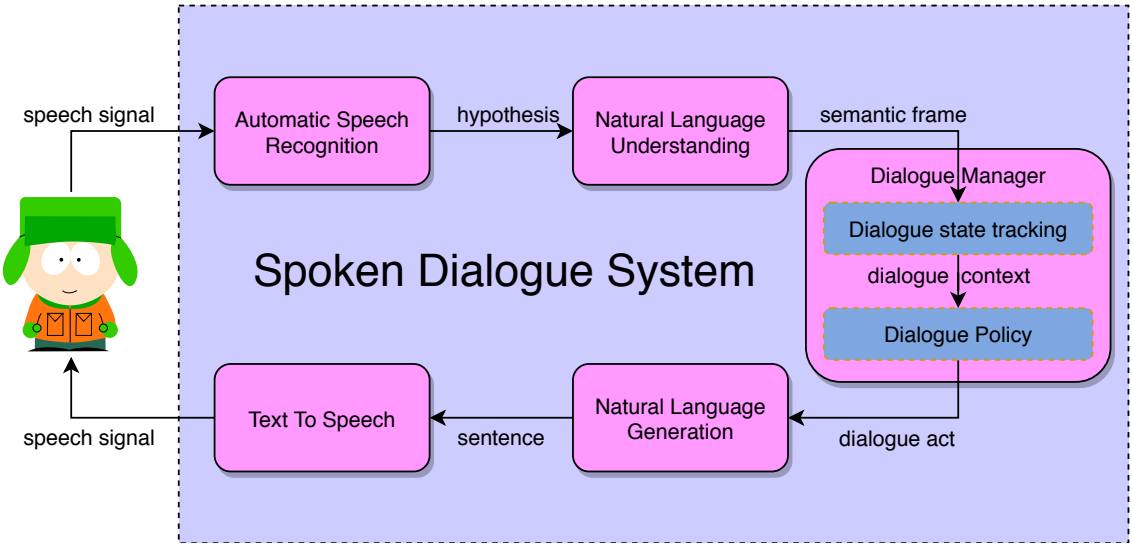


Figure 3.1: The architecture of a modular Spoken Dialogue System

et al. 2017). Recent solutions are discriminated between two approaches. The first one considers using NNs (Deng et al. 2012; Tür et al. 2012; Yann et al. 2014) or even Convolutional Neural Network (Fukushima 1980; Weng et al. 1993; LeCun et al. 1999) to directly classify the user sentence from a set a pre-defined intents (Hashemi 2016; Huang et al. 2013; Shen et al. 2014). The second one, put a label on each word of the user's sentence. Deep Belief Networks (Deng et al. 2012) have been applied (Sarikaya et al. 2011; Deoras et al. 2013). RNNs have been also used for slot-filling NLU (Mesnil et al. 2013; Yao, Zweig, et al. 2013) and LSTM (Yao, Peng, et al. 2014). Some approaches consider bringing together user intent and slot-filling (X. Zhang et al. 2016).

Dialogue Manager

By leveraging from the other modules, the DM can decide the best thing to say, given the current state of the dialogue. It is divided into two sub-modules, the Dialogue State Tracking (DST), and the Dialogue Policy (DP).

Dialogue State Tracking The DST keeps track of the history of the dialogue (what the user already asked, what slot is already filled, etc) and compiles it into a dialogue state (could be called a belief state or also a dialogue context, depending on the framework). It usually takes the form of confidence probabilities of each slot. Traditional approaches are rule-based (Goddeau et al. 1996; Sadek et al. 1997). Statistical methods use Bayesian networks as recalled in (Thomson 2013; Henderson 2015). A recent approach is to consider merging NLU and DST into a single RNN. The dialogue state is then the hidden layer of a RNN (or LSTM) supposed to infer the next word in the dialogue (T. H. Wen et al. 2017; Barlier et al. 2018a).

Dialogue Policy The next step involves the Dialogue Policy (DP), extensively described in Section 3.3, that chooses a dialogue act (Austin 1962; Searle 1969) according to the dialogue context. The most simple form of dialogue act is a parametric object used by the NLG module to reconstruct a proper sentence. For example, if the domain

of application is restaurant reservation, the DM may ask for the area of the restaurant using the following dialogue act: `request(slot=area)`[1]. In some recent work, the DP actually outputs words (J. Li et al. 2016; Vries et al. 2017). It seems natural to process this way, but that means the DM must learn the semantic of the language. Without proper metrics, it may lead to DMs optimising the task with incoherent or ill-formed sentences.

It is worth noting that in the literature, DST and DP are not necessarily exposed as two distinct modules. For example, in (S. Young et al. 2009), they cast the DST and the DP as a single POMDP. The embedded Bayesian network of the POMDP acts as the DST.

Natural Language Generation

The NLG module would transcript the dialogue act `request(slot=area)` as "Where do you want to eat?". Recently, LSTM for NLG has been used in the SDS context (T.-H. Wen et al. 2015).

Text To Speech

Finally, the TTS module transforms the sentence returned by the NLG module into a speech signal. State-of-the-art approaches use generative models (Van Den Oord et al. 2016; Y. Wang et al. 2017; Oord et al. 2018).

3.2 On the slot-filling problem

In this thesis, we more specifically address slot-filling dialogue problems. For instance, we may consider an online form to book train tickets. The form contains several slots to fill, as for example birthdate, name, and address. The regular approach consists in filling each slot manually then send the HTML form to a server. This method exists for decades and has been used extensively on websites. The advantage of this approach is that it is exact since forms include checkboxes and radio buttons and all other inputs are filled according to their labels (name, address, etc). The counterpart is that the filling procedure may feel cumbersome to the user. Interacting with the form directly through voice or chat instead of filling each slot may ease the process and this is the approach we consider. It involves a DS asking the user the value of each slot in order to fill the form, then return the result of the form to the user, partially or entirely, depending on the user request. The user experience is enhanced as the user interacts in a natural fashion with the machine. Also, it can speed up the process as the user can provide several slot values in a single utterance.

3.2.1 Settings

In order to describe the slot-filling problem, we use a taxonomy similar to the Dialogue State Tracking Challenge (DSTC) taxonomy (Jason D. Williams et al. 2013). The problem is, for the system, to fill a set of slots (or goals). A slot is said *informable* if the user can provide a *value* for, to use as a constraint on their search. A slot is said *requestable* if the user can ask the system the value of a slot. All *informable* slots are also *requestable*. In the book train tickets example, the departure city is an *informable* with a number of possible *values* equal to the number of cities served by the transport. A non *informable* slot, but *requestable*, would be, for example, the identification number of the train.

[1] More details are provided in Section 3.2

User and system acts

For purposes of conducting the dialogue, the user and the system are given respectively a set of user acts and system acts. Some acts are common to every dialogues such as `hello`, `repeat` or `bye`. Others acts depend on the ontology of the domain as they are direct instances of the *requestable* and *informable* slots. Both actors can request the value of a slot using the generic act `request(slot="a slot")` and inform the value of a slot using `inform(slot="a slot", value="its value")`. The counterpart of the DS slot-filling procedure is that an utterance may be misunderstood by the system. As the system is given an NLU or NLU score, it is able to judge if an utterance is worth asking for repeating. The system can ask the user to repeat with several system acts:

- `repeat`: the user may repeat the last utterance. For instance "I don't understand what you said, please repeat.".
- `expl-conf(slot="a slot", value="its value")`: the system requests the user to explicitly acknowledge a slot value. For example "You want to book a train departing from Paris, is it correct ?".
- `impl-conf(slot="a slot", value="its value")`: the system reports a slot value without explicitly asking the user to confirm it. If the user thinks the value is wrong, then he can decide to fix the mistake. For example "You want to book a train departing from Paris. Where do you want to go ?".

Note that, in Part II and Part III, experiments will be conducted on slot-filling problems with a similar taxonomy, but the acts may differ.

3.3 The Dialogue Manager

The particularity of the DM as opposed to the other SDS's modules, is that it is stateful, in the sense that it needs to keep track of the dialogue state to operate optimally[2]. For example, we do not want to ask the same question twice if we have already got a clear answer. The DST is stateful by definition but most of the time the DP is stateless (it doesn't keep track of a state). This thesis proposes solutions to optimise the DM.

Since the DM is the only module involved, we abstract all the remaining modules into a single object called environment. Figure 3.2 describes the simplified workflow: we assume the DM to receive an object o called the observation. It contains the last user dialogue act a_{usr} (the semantic frame) and the SRS. We assume the DST updates the next dialogue state s' given the current state s, the last system act a_{sys} and the last observation o. In this thesis, the DST outputs a simple concatenation of the previous observations and system acts. Also, we restrict the system acts to dialogue acts only (and not words).

Traditionally, handcrafted approaches have been considered to design the DP. They just match the dialogue state to a dialogue act using a set of handcrafted rules for the DP. It has been shown to be unreliable in unknown situations and it requires an expert knowledge on the domain. Statistical methods may solve these issues. Generative models have been considered to predict the next dialogue act given the current state. This is typically how chit-chat bots work (Serban, Sordoni, et al. 2016; R. Yan 2018; Gao et al. 2019). State-of-the-art statistical approaches for task-oriented DSs involve RL.

[2]With modern approaches, any module using a RNN may be also considered stateful.

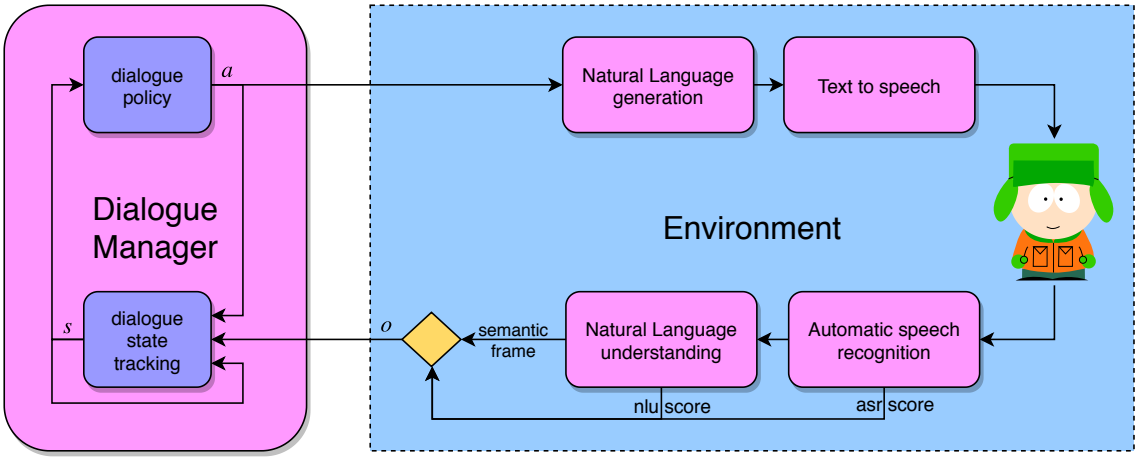


Figure 3.2: The pipeline view from the Dialogue Policy

A Reinforcement Learning problem

The previously cited methods do not really capture the sequential essence of a dialogue. By inferring the next dialogue act, they just plan one step ahead. In order to plan multiple steps ahead, one can cast the DP as a sequential decision making problem. Markov Decision Processes (MDPs) are a good fit to mathematically describe those problems but they require a model of the environment. Unfortunately, it is not possible to get an analytic model of the environment since it involves human decision making. To overcome this issue, dialogue policies can be optimised using Reinforcement Learning (RL) algorithms, where the only prerequisite is having a dialogue corpus or the ability to directly interact with the user in an Online fashion.

4. Training the Dialogue Manager with RL

In the previous chapter, we saw that a fully-operational Dialogue System (DS) is a complex pipeline including several sub-modules. But, this thesis being focused on the Dialogue Manager (DM), we will abstract all the remaining DS modules, and the user, into a single entity called the environment. The training of a DS will hence simplify, without loss of generality, into the optimisation of a Dialogue Policy (DP) interacting with this "augmented" environment. From now on, we state an equivalence between an environment and a user and just forget all the other modules.

Usually, in dialogue applications, particularly with task-oriented dialogue applications, the system must keep track of the history of the dialogue. If we keep the whole dialogue history in the dialogue state s_i, the DP has all the information it needs to take the better decision at time i in order to complete the dialogue task: the dialogue state representation is said *Markovian*; more formally, the past states have no additional information to predict the future. Then, in order to keep track of the dialogue, the DM must update its state according to the previous state and the current observation raised by the dialogue using the Dialogue State Tracking (DST) module. Considering that an observation is conditioned on the current state and the dialogue act, then we have enough elements to represent the dynamics of the environment, denoted as the transition kernel P. Knowing that taking an action given a state leads to another state is helpful to predict multiple steps ahead *i.e.* which state the DP may reach if it follows a series of dialogue acts. However, it will not suffice to evaluate the policy. To that end, we add a new signal, the immediate reward function, denoted as R, that indicates how good was a dialogue act transitioning from one state to another. By compiling all those information, we are able to optimise the DP using a mathematical framework called Markov Decision Process (Bellman 1957).

Markov Decision Processes

We first introduce the core concepts used in this manuscript.

Definition 4.0.1 A Markov Decision Process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$ where:

- $\mathcal{S}$ is the state set,
- $\mathcal{A}$ is the action set,
- $R \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is the reward function,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S} \times \mathcal{A}}$ is the transition kernel; $\mathcal{M}(\mathcal{X})$ denotes the probability measure over a set $\mathcal{X}$.
- γ is the discount factor.

In the dialogue context, it may be hard to define a good reward function. For example in chit-chat applications, how to know if a conversation went well for the user? Knowing this answer implies using sentiment analysis which is not reliable in the current state-of-the-art. Manual labeling may work, but it requires a costly human labor. Even in those cases, one cannot really decide whether a user enjoyed the conversation without asking explicitly users to give feedback[1].

For task-oriented situation, if the task is completed, then the environment should yield a reward (usually discounted with respect to the length of the dialogue). In simulation, it is easy to know if the task is completed, but in real application, the agent might understood wrongly the request; a lot a work has been done to takcle this issue (El Asri 2016).

The state of an MDP must contain all past observations. In other words, the agent would not be able to take a better decision with additional past information to complete the dialogue task. We say that the process respects the Markov property (or that it is *Markovian*).

Definition 4.0.2 Let $(\Omega, \mathbb{P})$ a probability space. Then a stochastic process $X = \{X_i\}_{i \in N}$ is said to possess the Markov property if

$$\mathbb{P}(X_i = x_i | X_{i-1} = x_{i-1}, \ldots, X_0 = x_0) = \mathbb{P}(X_i = x_i | X_{i-1} = x_{i-1}). \tag{4.1}$$

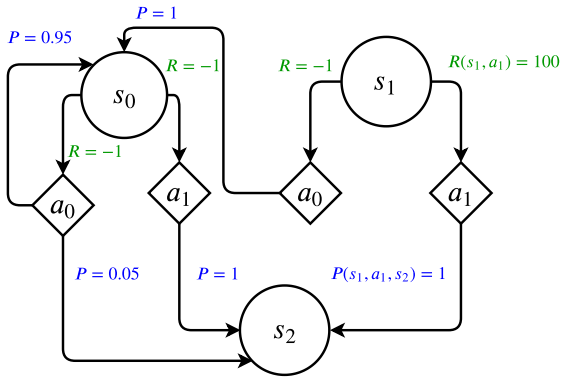
In the context of an MDP, that means:

$$P(s_i, a, s_{i+1}) = \mathbb{P}(s_{i+1} | s_i, a) = \mathbb{P}(s_{i+1} | s_i, \ldots, s_0, a). \tag{4.2}$$

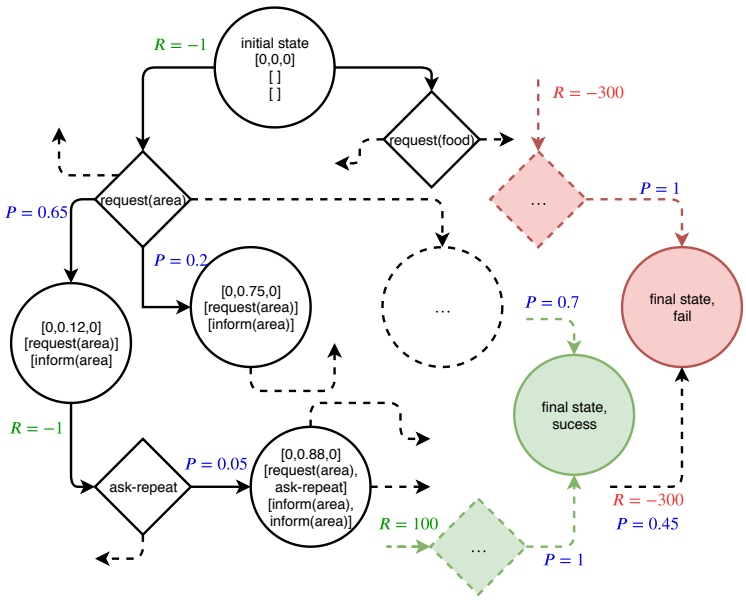
For example, if the agent is a car, in order to decide how to slow down when it faces another car, it must know its current speed, its position and the position of the other car. If the agent only knows its current position and not all other past positions, it will not be able to know its speed and the process will not be *Markovian*.

Figure 4.1a shows an example of a generic MDP illustrating Definition 4.0.1 side by side with Figure 4.1b, a partial view of an MDP from a slot-filling task-oriented dialogue application. The particularity of an MDP in the dialogue context is that the state space is actually infinite as the Speech Recognition Score (SRS) is a continuous variable. We also note that by keeping all the past dialogue acts, user acts and the last SRS for each slot in the dialogue state, we respect the Markov property. Indeed, there is no interest in keeping all the SRS from previous utterances as the only thing that matters: whether the system understood correctly a given slot. However, keeping all the past SRSs may be useful in a situation where the dialogue act includes the value of the slot. For example,

[1] Even with explicit surveys, there are several biases that must be corrected.



(a) A generic discrete Markov Decision Process. The state s_2 is final.



(b) A Markov Decision Process in a slot-filling dialogue task. Dashes indicate potentially infinite possibilities of states, transitions and actions.

Figure 4.1: Example of Markov Decision Processes

if the user responds to a `request(area)` by two `inform(area=Paris)` with SRS 0.7 and 0.9 and one `inform(area=London)` with SRS 0.3, it seems probable that the actual informed area is Paris.

The behaviour of the dialogue agent, the DP, is then defined as the policy $\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}$, that maps states to actions, which can either be deterministic or stochastic. Solving an MDP consists in finding a policy π^* that maximises the amount of rewards gathered by walking on the MDP. Usually the objective is to maximise the infinite-horizon γ-discounted sum of rewards, called return, in expectation:

Definition 4.0.3 Let $\gamma \in [0, 1[$, then the γ-discounted return for the policy π is

$$G^{\pi} = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i) \tag{4.3}$$

where $s_{i+1} \sim P(s_i, a_i, s_{i+1})$and $a_i \sim \pi(s_i)$.

From a task-oriented dialogue perspective, it may be strange to consider intermediate rewards. After all, the sole completion of the task matters, but usually, we also want to solve the task as fast as possible. For that purpose, there are two solutions to the agent designer (not mutually exclusive): yield a negative reward at each turn of the dialogue or set a γ strictly smaller than 1 to account for the hazard of the user terminating the dialogue at each turn (Fedus et al. 2019). In both cases, the longer the dialogue lasts, the lower the return will be. One may notice that anyway, for most of the following RL theory to apply, one needs the condition $\gamma < 1$.

From the return, one can derive the action-value function, also known as Q-function. This function yields the expected return after taking an action in a given state and thereafter following a fixed policy π.

Definition 4.0.4 Let $a \in \mathcal{A}$ and $s \in \mathcal{S}$ and π a policy, then

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G^{\pi} | s_0 = s, a_0 = a]. \tag{4.4}$$

Thanks to the Bellman Evaluation equation, we are able to compute the expected return of a policy using a simple linear system of equations:

Proposition 4.0.1 Let π be a policy, then $\forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$.

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} [P(s, a, s') \sum_{a'} \pi(a'|s') Q^{\pi}(s', a')]. \tag{4.5}$$

Bellman Evaluation equation can be reformulated as $Q^{\pi} = \mathcal{T}^{\pi} Q^{\pi}$ where $\mathcal{T}^{\pi}$ is the Bellman Evaluation operator. Being able to evaluate a policy is one thing, but what we really want is finding the best policy w.r.t. the expected return. We call optimal policy π^*, a policy that maximises the return in expectation *i.e.*

Definition 4.0.5 The policy π^* is said optimal if and only if

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a). \tag{4.6}$$

We now present the Bellman Optimality equation (or control equation) (Bellman 1956):

Theorem 4.0.2 Bellman Optimality equation: It exists a unique function, denoted as Q^*, that verifies the Bellman Optimality equation:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} [P(s,a,s') \max_{a' \in \mathcal{A}} Q^*(s',a')]. \tag{4.7}$$

Bellman Optimality equation can be reformulated as $Q^* = \mathcal{T}^* Q^*$ where Q^* is the optimal Q-function and $\mathcal{T}^*$ the Bellman Optimality operator (also know as dynamic programming operator). An important property is that we can construct an optimal policy as an expression of Q^*:

Proposition 4.0.3 The policy defined by

$$\pi^*(s) \in \arg\max_{a \in \mathcal{A}} Q^*(s,a) \tag{4.8}$$

is optimal.

All those results allow us construct algorithms that find this optimal policy.

Solving with dynamic programming

If $\gamma < 1$, the operator $\mathcal{T}^*$ is a contraction. Thanks to the Banach theorem (Banach 1922), it admits a unique solution. Then, one can find Q^* by iterating on Equation (4.7): the algorithm is called Value Iteration (VI) (Bellman 1957) and recalled on Algorithm 1.

Algorithm 1: Value-Iteration

Data: an MDP $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$

$k \leftarrow 0$
$Q_k \in \mathbb{R}^{|\mathcal{S} \times \mathcal{A}|}$
while $k = 0 \vee ||Q_{k+1} - Q_k|| < \upsilon$ **do**
  $Q_{k+1} \leftarrow \mathcal{T}^* Q_k$
  $k \leftarrow k + 1;$
end
$\pi(s) \in \arg\max_{a \in \mathcal{A}} Q_k(s,a) \; \forall s \in \mathcal{S}$
return π

Another approach is to evaluate the policy with Equation (4.5), then improve it greedily over the current Q-function of the policy. The algorithm is called Policy Iteration (PI) (Howard 1960) and is described in Algorithm 2. Theorem 4.0.4 ensures that the algorithm converges to the optimal policy. PI and VI are quite similar. One difference is that VI does evaluation and policy improvement at the same time. The mattering difference

is the stopping criterion: PI stops once the policy has converged which happens earlier than the convergence of the Q-function in VI.

Algorithm 2: Policy-Iteration

Data: an MDP $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$

$k \leftarrow 0$
$Q_k \in \mathbb{R}^{|\in \mathcal{S} \times \mathcal{A}|}$
π_k any policy
while $k = 0 \vee \pi_{k+1} \neq \pi_k$ **do**
  $Q_{k+1} = \mathcal{T}^{\pi_k} Q_k$ (evaluation)
  $\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} Q_{k+1}(s, a)\ \forall s \in \mathcal{S}$ (improvement)
  $k \leftarrow k + 1$
end
return π_k

Theorem 4.0.4 (Policy Improvement Theorem): Let π a policy, Q^π its associated Q-function and π' defined as :

$$\forall s \in \mathcal{S}, \pi'(s) \in \arg\max_{a \in \mathcal{A}} (R(s, a) + \gamma \mathbb{E}_{s'}[P(s, a, s') \max_{a' \in \mathcal{A}} Q^\pi(s', a')]) \tag{4.9}$$

then $\forall (s, a) \in \mathcal{S} \times \mathcal{A},\ Q^{\pi'}(s, a) \geq Q^\pi(s, a)$

with $Q^{\pi'}(s, a) = Q^\pi(s, a)$ iif $Q^\pi(s, a) = Q^{\pi^*}(s, a)$.

On the continuous state-space problem

As we already mentioned, the state-space $\mathcal{S}$ in the dialogue context is actually infinite since the SRS is a continuous variable. The tabular solutions of VI and PI are then intractable. To overcome this issue, we build an approximation of the Q-function. Let $\mathcal{F}$ be the set of representable functions with domain $\mathcal{S} \times \mathcal{A}$, $||\cdot||_\infty$ the uniform norm, and Ξ the projection operator onto $\mathcal{F}$:

$$\Xi f = \arg\min_{\widetilde{f} \in \mathcal{F}} ||f - \widetilde{f}||_\infty. \tag{4.10}$$

This process is known as function approximation and is heavily used in today RL algorithms. We can naturally extend VI to Fitted-Value-Iteration (FVI) (or Approximate Value Iteration (AVI), Bellman and Dreyfus (1959)) using a composition of the optimal Bellman operator with the projection operation. In a nutshell, the algorithm iterates over the following contraction[2]: $Q^* = \Xi \mathcal{T}^* Q^*$. Projection is usually done by sampling $\widetilde{\mathcal{S}} \subset \mathcal{S}$ a finite subset of $\mathcal{S}$ and then process a regression over the learning examples $\widetilde{\mathcal{S}} \times \mathcal{A}$. Let $\Gamma(X, Y) :\rightarrow \mathcal{F}$ denotes the regression algorithm, then the update rule is as follows:

$$Q^* \leftarrow \Gamma(\widetilde{\mathcal{S}} \times \mathcal{A}, \{\mathcal{T}^* Q^*(x)\}_{x \in \widetilde{\mathcal{S}} \times \mathcal{A}}). \tag{4.11}$$

[2]if the projection is considered perfect, i.e has no bias.

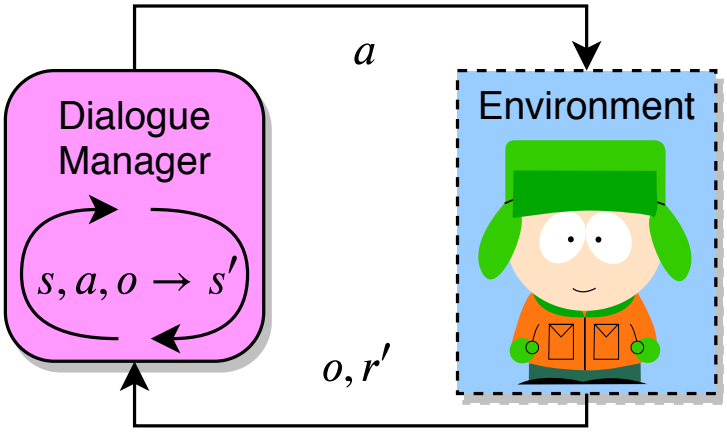


Figure 4.2: The Dialogue Manager cast as a Reinforcement Learning problem.

If we apply the same idea to PI (on the evaluation only), we obtain Approximate Policy Iteration (API) (Bertsekas 1996). Please note that the regression algorithm can take various forms, as for example linear regression, regression trees or even Neural Networks.

Toward Reinforcement Learning solutions for Dialogue Systems

Another problem arises when dealing with dialogue applications. The function R and P are actually determined by the user (and the plugged pipeline) the system dialogues with. That means that R and P correspond to a model of the user, which is complex, unknown and varies a lot from one individual to another. So in a the dialogue context, R and P must be learnt (or implicitly learnt) using information on the user. RL allows to handle this flow of information. RL is a framework used to describe an agent (e.g. the DM) interacting with an environment (e.g. the user and peripheral modules) in a sequential fashion. The DS pipeline is cast as an RL problem in Figure 4.2. The information contained in an interaction can be compiled into a single object called transition. This is a tuple of the form: (s, a, s', r') where s is the state of agent, a the action applied on the environment, r' the reward received from the environment and s' the next state of the agent. A batch $\mathcal{D}$ (a dialogue corpus) is a collection of N transitions: $\mathcal{D} = \{(s_i, a_i, s'_i, r'_i)\}_{i \in [0,N[}$.

Two classes of problems arise depending on how this information is gathered: on one hand, in Online RL, the objective is to learn on-the-fly a policy using the flow of transitions. On the other hand, in Offline (or Batch) RL, the objective is to directly learn a policy given a batch $\mathcal{D}$. In the next sections, we introduce algorithms from both categories that have been successfully applied to DM. We start with Batch RL.

4.1 Assuming a given dialogue corpus

Dialogue applications involving statistical methods usually assume that the dialogue corpus is given. A panel of dialogue datasets are available for research. To name the famous ones, there is the Ubuntu dialogue corpus (Lowe, Pow, et al. 2015) and the DSTC

dataset (Jason D. Williams et al. 2013). Please refer to Serban, Lowe, et al. (2015) for an extended overview. Usually, the challenge in using dialogue corpora for RL context is the design of the reward, but this is not the focus here in the thesis, so we assume having a well-formed dialogue corpus; as stated earlier, we denote it as a batch of transitions: $\mathcal{D} = \{(s_i, a_i, s'_i, r'_i)\}_{i\in[0,N]}$. In model-based approaches, we explicitly learn the unknown functions P and R using a regression algorithm and $\mathcal{D}$ as learning batch, then apply a planning algorithm (Lison 2013; Peng et al. 2018). In model-free batch RL, we do not explicitly estimate R and P but we can, among other methods, bootstrap the estimation of Q with the current reward r and the expected reward in the next state s'. For example, in Fitted-Q (FTQ) (Ernst et al. 2005; Riedmiller 2005), in addition to the projection operator, we apply a sampled Bellman Optimality operator on the batch:

Definition 4.1.1 Let (s, a, s', r') be a transition, and $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a function. The Sampled Bellman Optimality operator $\hat{\mathcal{T}}^*$ is :

$$\hat{\mathcal{T}}^*(f(s,a)) = r' + \gamma \max_{a'\in\mathcal{A}} f(s', a'). \tag{4.12}$$

The general FTQ algorithm (Equation (4.13)) is then an interactive process over the composed operator: $Q^* = \Xi\hat{\mathcal{T}}^*Q^*$ with respect to $\mathcal{D}$.

$$Q^* \leftarrow \Gamma(\{s_i, a_i\}_{i\in N}, \{r'_i + \gamma \max_{a'\in\mathcal{A}} Q^*(s'_i, a')\}_{i\in N}). \tag{4.13}$$

As in FVI, the designer is free to choose the regression algorithm suitable to his problem. Trees (Ernst et al. 2005) and NNs (Riedmiller 2005) have been successfully combined with general FTQ. If using a Least Squares (LS) linear-regression, the Q-function can be rewritten as

$$Q_\theta(s, a) = \theta^\top \phi(s, a), \tag{4.14}$$

with ϕ the feature function and θ the parameter to find. Linear LS optimisation results in computing θ with a simple matrix inversion. Let $M = (\sum_{i=0}^{N} \phi(s_i, a_i)\phi((s_i, a_i))^\top)^{-1}$, then FTQ is the following iterative process:

$$\theta_k \leftarrow M \sum_{i=0}^{N} \phi(s_i, a_i)(r'_i + \gamma \max_{a\in\mathcal{A}}(\theta_{k-1}^\top \phi(s'_i, a))). \tag{4.15}$$

The algorithm stops if $k \geq K$ where $K \in [0, \infty[$ the number maximum of iterations, or if $||\theta_k - \theta_{k-1}|| \leq \upsilon$. This solution has been adopted in (Pietquin et al. 2011). The authors chose an LS linear-regression with Gaussian radial basis functions as features in order to optimise a DM.

We can extend PI using the same idea applied to the Bellman Evaluation operator. The algorithm is called Least Squares Policy Iteration (LSPI) (Lagoudakis et al. 2003) and has been applied to dialogue tasks (L. Li, Jason D Williams, et al. 2009; Barlier et al. 2018b).

4.2 Online interactions with the user

Creating a fixed dialogue corpus is a difficult task. A dialogue is by essence an interactive and evolving process, and the existing corpora may be too expensive or even not suited

for the problem at hands. In consequence, in some situations, it may be of interest to simply learn, or adapt, on-the-fly the DM strategy while it dialogues with an unknown user/environment. In this section, we present algorithms to do such interactions with the user (Ferreira et al. 2013). Those methods belong to the class of Online RL algorithms.

The Exploitation/Exploration dilemma

For a new user, in order to gather the transitions the algorithm will learn with, the DM adopts decisions dictated by a DP π called the behavioural policy. In order to learn the optimal policy (target policy), π needs to gather all the transitions that would potentially be covered by the optimal policy. As we are actually building this policy, it is not possible to know in advance the useful transitions for learning. The basic idea to cover the state-action space as the optimal policy would do it is to perform an Exploration/Exploitation strategy. The exploration phase discovers new areas in the state-action space, while the exploitation phase directs the agent to a more profitable state-action space. Several implementations of the Exploration/Exploitation dilemma have been proposed, coming from the Bandit community most of the time. In this manuscript, we will use the notorious ε-greedy strategy which works as follows: when the agent is in state s, it chooses a random exploration action with probability ε, and it chooses the greedy action (the most profitable action according to some parameters, the Q-function for example) otherwise.

As long as π gathers transitions, the policy in construction (which is not necessarily π) tends to be a better approximation of the optimal policy. The agent must refine its knowledge around the profitable states and for that, the ε must decrease with respect to the number of transitions gathered. Exponential decay is commonly used for such a decreasing processes. The number ε is then a variable of i, the current number of transitions[3]: $\varepsilon(i) = \exp(-\lambda i)$ with λ dictating the vanishing speed. But, as the exponential decay procedure does not ensure convergence, it is usually safer to use a linear decay (Auer et al. 2002a): $\varepsilon(i) = \min\{1, 1/(\lambda i)\}$.

Note that we can already see the limitations of this approach for new users; it is not profitable to act randomly with a human discovering a new dialogue system, the risk of abandon is too high. We will see later that we can handle this issue with TL.

On-policy versus Off-policy

Online RL algorithms can be classified into two subcategories depending on the nature of the behavioural policy. If the algorithm improves this policy as it gathers transitions, then we say the algorithm is On-policy. If the algorithm optimises a policy distinct from the behavioural policy, then we say the algorithm is Off-policy.

The classic Off-policy algorithm is Q-learning. It updates the Q-function according to the Temporal Difference (TD) error. This error captures the difference between the current estimation of the expected return and the immediate reward summed to the expected return in the next state. The Q-learning is recalled in Equation (4.16). The hyper-parameter α is the learning rate and (s_i, a_i, s'_i, r'_i) is the last transition i the behavioural policy generated.

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha[r'_i + \gamma \max_{a' \in \mathcal{A}} Q(s'_i, a') - Q(s_i, a_i)]. \tag{4.16}$$

To ensure convergence, the learning rate must decrease linearly. Q-learning is not fit to learn the DP as it is a tabular solution and as we saw earlier, the dialogue state contains

[3] or trajectories, this does not matter.

continuous variables. In the same spirit of tractability in Offline RL, one can extend Online algorithms using function approximation.

Function approximation

A straightforward approach adopted for Q-learning is the LS linear-regression of a model of the Q-function using Gradient Descent (GD). The Q-function is approximated with a linear model: $Q(s,a) = \phi(s,a)^\top \theta$ where ϕ is the feature vector and θ the parameter vector to optimise. The optimiser is not a GD strictly speaking as we regress Q over the error of a single transition rather the whole batch. Indeed, the algorithm receives transitions one by one, so the optimiser actually used is the Stochastic Gradient Descent (SGD). The loss L is defined as:

$$L(\theta) = \frac{1}{2}(r'_i + \gamma \max_{a' \in \mathcal{A}} Q(s'_i, a') - Q(s_i, a_i))^2 \tag{4.17}$$

$$= \frac{1}{2}(r'_i + \gamma \max_{a' \in \mathcal{A}} \phi(s'_i, a')^\top \theta - \phi(s_i, a_i)^T \theta)^2. \tag{4.18}$$

We define $Q^+ = \gamma \max_{a' \in \mathcal{A}} \phi(s'_i, a')^\top \theta$. Then we compute the gradient of the loss considering Q^+ constant with respect to θ. This method is called semi-gradient and is developed as follow:

$$\nabla L(\theta) = \nabla \frac{1}{2}(r'_i + Q^+ - \phi(s_i, a_i)^\top \theta)^2 \tag{4.19}$$

$$= (r'_i + Q^+ - \phi(s_i, a_i)^\top \theta) \cdot \phi(s_i, a_i). \tag{4.20}$$

We update θ such that we minimise the loss. The update rule recalled in Equation (4.21) defines the LS Q-learning algorithm[4]:

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla L(\theta_k) \tag{4.21}$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha(r'_i + \gamma \max_{a' \in \mathcal{A}} \phi(s_{i'}, a')^\top \theta_k - \phi(s_i, a_i)^\top \theta_k) \cdot \phi(s_i, a_i) \tag{4.22}$$

Riedmiller (2005) proposed a natural extension to FTQ in the Online setting; It involves alternating between two phases: it generates a batch of transitions with an ε-greedy policy using the current target policy as the greedy component; Then it updates the target policy with all previous batches. It repeats this operation until ε decays enough.

Learning from scratch

In dialogue system applications, it is not usually easy to learn a DP from scratch using Online RL algorithms. Indeed, the first interactions with the user are crucials to build a bond between the service and the user; taking random actions, using ε-greedy, to gather information about the environment (including the humain) may just end-up with user dropouts. To tackle this problem, it is possible to contraint the action-space to get the relevant actions for a given state (Singh et al. 2002; Jason D Williams et al. 2008; Laroche, Putois, Bretier, and Bouchon-Meunier 2009; Laroche, Putois, and Bretier 2010). As those solutions necessitate an expertise on RL, new studies introduced techniques suited to an average developer; to name a few: RL results monitoring (Laroche, Bretier, et al. 2010), convergence speed prediction (El Asri and Laroche 2013) or interaction quality prediction (El Asri, Khouzaimi, et al. 2014). In this thesis we tackle this problem using Transfer Learning (TL), as we will see in the next chapter.

[4] In this context, $i = k$ as the parameters are updated for each new transition.

4.3 To go beyond

There exists a lot of solutions in RL applied to DM that goes beyond the scope of this thesis. Very popular approaches involve Deep Reinforcement Learning (DRL): action-value approximation with DQN (Mnih et al. 2015) used to solve "Settlers of Catan" a negotiation game (Cuayahuitl et al. 2015); Policy Gradient (PG) methods with REINFORCE (R. J. Williams 1992) to play the the multi-modal game GuessWhat?! (Vries et al. 2017); Actor-Critic methods with Actor-Critic-Experience-Replay (Z. Wang et al. 2016) have also been applied to the DM (Weisz et al. 2018).

5. User adaptation and Transfer Learning

In the previous chapter, we introduced Reinforcement Learning (RL) solutions for the Dialogue Manager (DM). For that, we cast the problem as an agent, the DM, interacting with an environment: the user. In classic approaches, we actually consider the environment as any potential user, and then learn a policy able to converse with this *generic* user. This approach assumes all users to adopt the same dialoguing behaviour, although this not necessarily the case. The designer may choose to represent all users as the same unique entity to accelerate the learning of the DM. It is a fair argument indeed, but it may impact negatively the overall quality of the dialogue. In consequence, in the following, we present solutions to overcome this limitation. We adopt the Transfer Learning (TL) framework (Lazaric 2012; Taylor and Stone 2009) where the objective consists in improving the learning of a target task, thanks to the knowledge obtained from a set of similar source tasks. In dialogue, we usually associate a task with a unique user or a domain. Then, the DM communicates with an unknown user/domain based on its past experience with other users/domains. One could also consider a setup where there is a single user with several communication channels making different tasks. For example, talking through the phone, VoIP, or face to face. That being said, in this manuscript, we consider one user by task.

5.1 The problem of Transfer Learning

As stated in Lazaric (2012), TL leverages the knowledge collected from a number of different tasks to improve the learning performance in new tasks. In our case, a *task* is defined as "dialoguing with a user u" and as we saw in previous chapter, modelled as an MDP. We denote this MDP as $u = \langle \mathcal{S}_u, \mathcal{A}_u, P_u, R_u \rangle$[1]. The TL vocabulary is the

[1]By abusing the notations, the user u, its MDP and the task describe the same concept.

following[2]: the *domain* of the task u is the state-action space $\mathcal{S}_u \times \mathcal{A}_u$; The *space of tasks* is the set of potential users $\mathscr{U} = \{u_j\}_{j\in[0,M]}$; The *environment* $\mathscr{E} = \langle \mathscr{U}, \Omega \rangle$, is denoted as the the probability distribution Ω over the space of tasks. The environment $\mathscr{E}$ fully describes the problem: the DM must find the best strategy to adopt with a target user u_t drawn from the distribution Ω. In order to operate this learning quickly, the DM has access to the knowledge gathered previously with source users $\mathscr{U}_s = \{u_s\}_{s\in[0,M_s]} \sim \mathscr{E}$ drawn from the environment. In a sense, we can see the TL problem as a Supervised Learning (SL) problem where already encountered users stand for the learning base and the new user is part of the test base. The idea is then to construct a general model, that returns a DP given a user, which is able to generalise across unseen users. In what follows, the adjectives source/target may be combined to other notions. For example, a source policy π_s denotes a policy learnt with a source user u_s, the target transitions $\{(s_i, a_i, r'_i, s'_i)\}_{i\in N_t}$ denote the transitions gathered with the target user u_t.

Let $\mathcal{K}$ be the space of the knowledge spawned by all users. As we do not access to the whole space, the target user knowledge may be a part of it and this is where TL comes in. Two distinct phases constitute a TL algorithm: the *knowledge-transfer* phase that extracts from $\mathcal{K}$ the relevant subset of knowledge for the target task and the *learning* phase that constructs a policy accordingly. The process may be strictly sequential or cyclic. In the literature, TL approaches differ following three criterions: the type of knowledge transferred, the objectives, and the setting.

The type of the knowledge

Three types of transfer are reported in the general TL literature: the transfer of transitions where the interactions with previous tasks are used to learn the policy for a target task (Sunmola et al. 2006; Lazaric 2008; Taylor, Jong, et al. 2008); the transfer of representations, for example state features (Ferguson et al. 2006; Mahadevan et al. 2007; Ferrante et al. 2008), action space (Sherstov et al. 2005), some layers of a NN or temporal abstractions for policies (Sutton et al. 1999); the transfer of parameters, including meta parameters (learning rate, discount factor etc), Q-function initialisation (Torrey et al. 2005) or even source policy transfer (Taylor, Whiteson, et al. 2007).

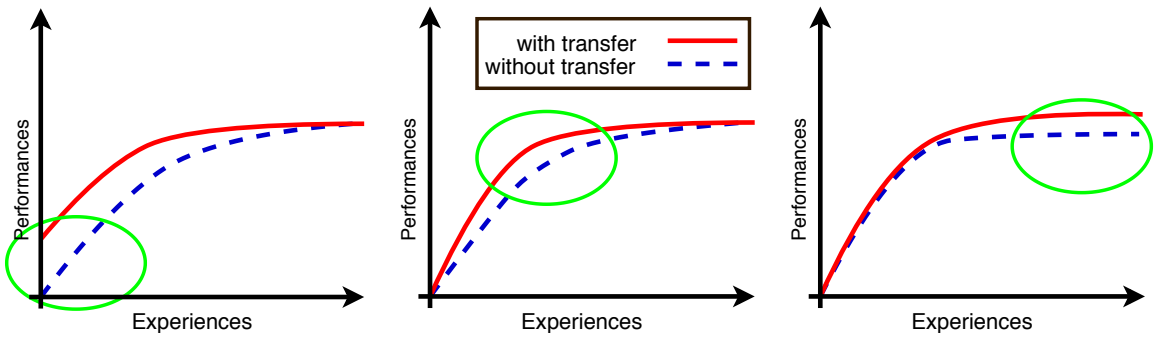
The objectives

In RL, we can distinguish three objectives that represent the performances of an agent: the jumpstart performance[3], where we measure how well an agent performs in the very first interactions with the environment (the user); the learning performance, where we measure how well the agent performs when improving its current policy; finally, the TL agent monitors an additional objective called asympotic performance, which measure the performances when the target dataset contains all the knowledge needed to compute the optimal policy.

The three objectives are displayed in the green circles of Figure 5.1. TL agent may optimise one, two or three of those objectives. A phenomenon we want to avoid, called negative transfer, is when the performances of an agent are better without transfer. A typical instance of this phenomena may occur when transferring transitions. Importing the source transitions may help to learn an initial generic policy improving jumpstart and

[2]The notation and the description of TL are largely inspired from the survey Lazaric (2012)

[3]Also know as bootstrap or coldstart performance.



(a) jumpstart performance (b) Learning performance (c) Asymptotic performance

Figure 5.1: Objectives of Transfert Learning (Langley 2006; Lazaric 2012)

learning performances, but if the transitions are too far from the target environment, and the learning algorithm has no mechanism to forget this knowledge, the target model will be biased and the asymptotic performance will be affected.

The setting

We distinguish TL given two setting properties:

- whereas the domain $\mathcal{S}_u, \mathcal{A}_u$ is fixed between the tasks. Usually, if the domain is not fixed, TL solutions focus on solving a simpler problem where the source task is unique. The difficulty lies in the mapping between those two tasks;
- whereas there is only one source task or several. The distinction may be hard to operate as one can consider, for example, a set of different users as a single *average* user, hence a single task. So to make things clear, here, we propose three settings:
  - *mono-task* where there is only one source task and one target task. The method is a direct mapping between those two tasks;
  - *generic-task*, where there is only one source task, but that may be several tasks seen as a generic task and there are potentially multiple target tasks;
  - *multi-task*, where there is several source tasks, and the distinction between tasks is made clear and relevant for the *knowledge-transfer* phase. There are potentially multiple target tasks.

  Please note that given this description, a method defined for a generic-task setting is not necessarily worse than a method defined for the *multi-task* purpose. On the contrary, methods relying on *mono-task* transfer are by definition limited to the tasks.

In this thesis, we propose solutions for user adaptation TL with *fixed domain* and we consider a *generic-task* and a *multi-task* setting.

5.2 State-of-the-art of Transfer Learning for Dialogue Systems

We distinguish two main parts in TL for DSs. A whole part of the literature tackles the problem of compatibility between different domains. Here the term domain stands for "domain of application" and not the domain $\mathcal{S}_u, \mathcal{A}_u$, even if it is closely related. For example, how to apply a policy learnt for a restaurant reservation application to a hotel reservation application. Please note that in this case, the tasks denote MDPs representing

different domains and not users (as explained in the previous section). This part will be discussed briefly. The second part of the literature concerns user adaptation, which will be further studied in the thesis. Finally, we will discuss a solution for evaluating real-life dialogues with prior knowledge. All methods discussed are listed in Table 5.1.

5.2.1 Cross domain adaptation

The first application of TL to DM has been proposed by (Gasic, Breslin, et al. 2013) and it was for cross domain adaptation.

Gasic, Breslin, et al. 2013

This approach considers extending a DP ability of handling several slots from a basic domain (the source environment), including *food-type* slot and the *area* slot for example, to an extended domain (the target environment) equipped with an additional unknown slot, the *price-range* slot. DMs are Gaussian Process (GP) DPs (Rasmussen 2003; Gasic and S. Young 2013) using a transitions dictionary to keep the problem tractable (Engel et al. 2006). The DST module is implemented using The Bayesian Update of Dialogue State (Thomson and S. Young 2010) for tracking belief states.

First, they propose a novel idea which allows a policy trained in the basic domain to be used in an extended domain: a kernel function that defines the correlation between belief states from different domains. Then, they investigate two solutions in order to quickly adapt to the new domain:

- Simply continue the training of the source policy in the extended domain. To that extent, a zero-mean prior is set on the GP for the Q-function. The *knowledge-transfer* phase consists in including transitions from both domains in the dictionary[4].
- The prior of the mean of the GP Q-function in the extended domain is the posterior of the mean from the GP Q-function in the basic domain.

The authors validate both approaches with human experiments showcasing significant improvements during the jumpstart phase.

To go beyond

Other approaches that go beyond the scope of this thesis have been released in the past few years. In Keizer and Rieser (2016) and Keizer and Rieser (2018), the authors solve the cross-domain problem by casting the DM as a multidimensional system. They learn common transferable skills.

A multi-agent solution is adopted in L. Chen et al. (2018). They learn two types of agent, *slot-dependent* and *slot-independent* agents. Rather than transfering the common knowledge as in Keizer and Rieser (2018), they transfer *slot-dependent* agents in the target domain.

In Ilievski et al. (2018), they propose to operate transfer using the weights of the policy's NN from the source domain to the target domain. But because domains are not the same, they have to extend the dialogue state and action spaces of both domains in order to render the transferred knowledge compatible.

[4] It is quite hard to categorise the type of knowledge transferred as GP are nonparametric, so the transitions represent directly the parameters. But following the author description, it is policy transfer.

5.2.2 User adaptation

When the different users denotes the space of tasks, we talk about user adaptation. Before listing the state of the art in user adaptation, we recall why it is essential.

On the need of user adaptation

Dialogue Managers (DMs) are usually trained versus user-models (Eckert et al. 1997; Levin and Pieraccini 1997; Levin, Pieraccini, and Eckert 2000; Pietquin 2004; El Asri, He, et al. 2016) , agenda-based user-simulator (Schatzmann 2008; Keizer, Gasic, et al. 2010) or dialogue simulator (Laroche 2017; Khouzaimi et al. 2017). Initially, models were trained and tested against the same user-model.

In Schatztnann et al. (2005), they demonstrated empirically that testing DPs trained on poor user-model (the bigram model), against more sophisticated user-models (the Levin-model or the Pietquin-model) led to poor performances. Assuming humans are more complex that the finest user-model, it is natural to infer that DM learnt on user-models may perform poorly in real conditions. Even if the paper focused on the quality of user-models, we see that any DM cannot be used for any target user-model (or human), thus a *knowledge-transfer* phase is essential. The same idea applies if the target environment is a parametric user-model simulating different kind of users. In Lemon and X. Liu (2007), they learnt different policies against user-models with 2 types of variations: user type (Cooperative/Uncooperative), and noise conditions (High/Low). Even if the policies trained in high-noise conditions generally perform better than those trained for low-noise conditions, no policies is the best fit for all user-models, thus a *knowledge-transfer* phase is also needed. While those two examples are not TL strictly speaking, they clearly show that user adaptation is essential. There are indeed a handful of published works in this domain that we survey in the following paragraphs.

Casanueva et al. 2015a

In this paper, they propose to personalise DSs to different speakers suffering from dysarthria. Users are simulated and their respective dysarthia seriousness is adjusted using the ASR module. As in Gasic, Breslin, et al. (2013), GPs are GP policies; they are learnt with Deterministic Training Conditional. In order to adapt the target policy to new users, they introduce two *knowledge-transfer* solutions which adress respectively two issues: which source to transfer from and how to weight the transitions transferred from multiple sources.

To select the source to transfer, they introduce a similarity function between speakers (along with 3 different speaker features) which is a Radial Basis Function kernel. They also pick the most similar transitions to the target speaker to reduce the amount of transferred data.

In order to weight the transitions transferred, they reduce the covariance of two transitions where the source and target speakers are similar. They make it possible by extending the transitions with the speaker features. Then, they extend the GP kernel defined in the state-action space with the Radial Basis Function kernel in the speaker space previously mentioned. The authors show that this method outperforms the source selection.

To design the target policy[5], they propose to divide the transferred transitions in two

[5]Please note that there is no *learning* phase strictly speaking since GP ML algorithms use lazy learning (instead of learning a model with the data, then doing inferences with the model, the data is directly used during the inferences).

sets. The first set allows one to learn the posterior of the mean of the source GP Q-function used as the prior in the target context, which is one of the solutions proposed in Gasic, Breslin, et al. (2013). The second set is used to initialise the set of points needed to compute the parameters of the target GP Q-function.

Genevay et al. 2016

The method proposed in this paper embraces the problem of user adaptation by transferring the transitions of previous encountered users to learn the target policy with FTQ. An algorithm is also proposed to pick the source set of transitions using Upper Bound Confidence (UCB). Finally, Density-Based, a way of picking the relevant transitions in the source set for the target task, is also introduced. The overall framework is tested on the Negociation Dialogue Game (NDG). The main limitation is the following: as the number of source user grows, the number of arms for the UCB algorithm grows too. It becomes impracticable to test at least once each arm with the target user. One of the contributions we propose in this thesis fixes this problem, hence for more details, please refer to Chapter 6.

Carrara, Laroche, and Pietquin 2017

In this work, detailed Chapter 6, we extend Genevay et al. (2016) in order to handle a larger pool of source policies. We propose a clustering approach and test it on handcrafted users as well as human-model users.

Mo et al. 2018

The approach introduced in this work, called PETAL, transfers parameters to process user adaptation with a real-world coffee ordering dataset. To that extent, they model the Q-function of the target policy as the sum of a common Q-function learnt using data gathered on several source users and a personalised Q-function learnt on the target transitions. A weight vector w representing the influence of the common Q-function over the personalised Q-function is extracted from the same data. Finally, they construct a DST module that is a function mapping dialogue history to belief state using a state-projection matrix denoted M. This matrix is also learnt with the source transitions. In the end, they transfer three parameters set to learn the target policy: the parameters of the common Q-function, the matrix M and the vector w. The authors use a State–Action–Reward–State–Action (SARSA) (Rummery et al. 1994) algorithm to regress the target Q-function. They validate the approach on both simulated and real work data comparing several baselines (including Casanueva et al. (2015a), Genevay et al. (2016) and Gasic, Breslin, et al. (2013) modified for user adaptation).

Carrara et al. 2018c

We introduce an approach based on the transfer of a safe policy learnt on the complete pool of source users. This solution is discussed in Chapter 8.

5.2.3 An aside on dialogue evaluation

On real SDS problems, it is usually hard to infer on-the-fly if the dialogue ended well. For example, in a task-completion problem, during the dialogue, the system cannot know if a slot has been understood thus it is impossible to infer if the task is a success or not. If the recognition score is high, there is more chance that the system understood correctly, but still, it is not a perfect measure. To that extent, El Asri et al. (2014) proposes to transfer a

	Knowledge	Setting	Metric
cross domain			
Gasic, Breslin, et al. 2013	parameter (transitions/policy)	mono-task cross-domain	all
Keizer and Rieser 2018	parameter (policy)	generic-task cross-domain	all
L. Chen et al. 2018	parameter (policy)	generic-task cross domain	all
Ilievski et al. 2018	parameter (policy)	mono-task cross-domains	all
user adaptation			
Casanueva et al. 2015a	parameter (transitions/policy)	multi-task fixed domain	all
Genevay et al. 2016	transitions parameter (policy)	multi-task fixed domain	all
Carrara, Laroche, and Pietquin 2017	transitions parameter (policy)	multi-task fixed domain	all
Mo et al. 2018	parameter (Q-function, DST)	generic-task fixed domain	all
Carrara et al. 2018c	parameter (policy)	generic-task fixed domain	all
dialogue evaluation			
El Asri et al. 2014	representation (reward function)	generic-task fixed domain	all

Table 5.1: Transfer Learning for Dialogue Systems

reward function inferred from a set of evaluated dialogues (by experts or other evaluation techniques). The authors learn this function using Inverse Reinforcement Learning with the reward shaping algorithm proposed in El Asri et al. (2012). They learn the target policy using LSPI on a target environment equipped with the learnt reward function.

II Scaling up Transfer Learning

6 A complete pipeline for user adaptation 55

6.1 Motivation
6.2 Adaptation process
6.3 Source representatives
6.4 Experiments
6.5 Related work
6.6 Conclusion
6.7 Discussion

6. A complete pipeline for user adaptation

6.1 Motivation

In this chapter, we tackle the problem of fast optimisation of user-adapted dialogue strategies by means of Reinforcement Learning (RL) as stated in Chapter 5 . The main goal is to improve jumpstart learning of RL-based dialogue management strategies when facing new users, by transfering data collected from similar users (Lazaric et al. 2008). To do so, we consider the setting in which a large amount of dialogues have been collected from several users, and a new user connects to the service (Genevay et al. 2016). This solution combines techniques from the literature on Multi-Armed Bandit (MAB) (Auer et al. 2002b), batch RL (Ernst et al. 2005; L. Li, J. Williams, et al. 2009; Chandramohan, Geist, and Pietquin 2010; Pietquin et al. 2011) and policy/MDP clustering (Chandramohan et al. 2012; Mahmud et al. 2013).

Instead of clustering user behaviours as in Chandramohan et al. (2012), we propose to cluster the policies that are trained on the user dialogue datasets. To do so, we define a novel policy-based distance, called PD-DISTANCE. Then, we investigate several clustering methods: k-medoids (Kaufmann et al. 1987) and k-means (Steinhaus 1957; MacQueen 1967), which enable the identification of *source representatives* for the Transfer Learning (TL). Once clusters representatives have been selected, they are plugged into a multi-armed bandit algorithm, as proposed in Genevay et al. (2016).

Following previous work where user adaptation (Janarthanam et al. 2010; Ultes, Kraus, et al. 2015) was used to address negotiation tasks (Sadri et al. 2001; Georgila et al. 2011; Barlier et al. 2015a; Genevay et al. 2016), we test our methods on different types of users involved in a negotiation game (Laroche and Genevay 2017). Methods are compared with two baselines: learning without transfer and transfer from a generic policy learnt from all the sources. These methods are tested by interacting with handcrafted users and human-model users learnt from actual human interactions (unlike Genevay et al. (2016)). These experiments show that our clustering methods provide a better dialogue experience

than the generic methods in both setups.

We present the full user adaptation process in Section 6.2. The clustering methods are described in Section 6.3. Section 6.4 describes the negotiation game and the experiments.

6.2 Adaptation process

Figure 6.1 shows the full process of user adaptation. As an input, we assume the existence of a database of dialogues with different source users, which allows the training of user specialised policies. At first, the process consists in searching or constructing policy representatives for this database so as to reduce the number of possible transfer sources. This is where the contribution of this chapter mainly stands, the rest being mostly inherited from Genevay et al. (2016).

6.2.1 The *knowledge-transfer* phase

The *knowledge-transfer* phase is operated on two levels: the selection of a source policy (system) and the selection of the relevant transitions used to learn this source policy.

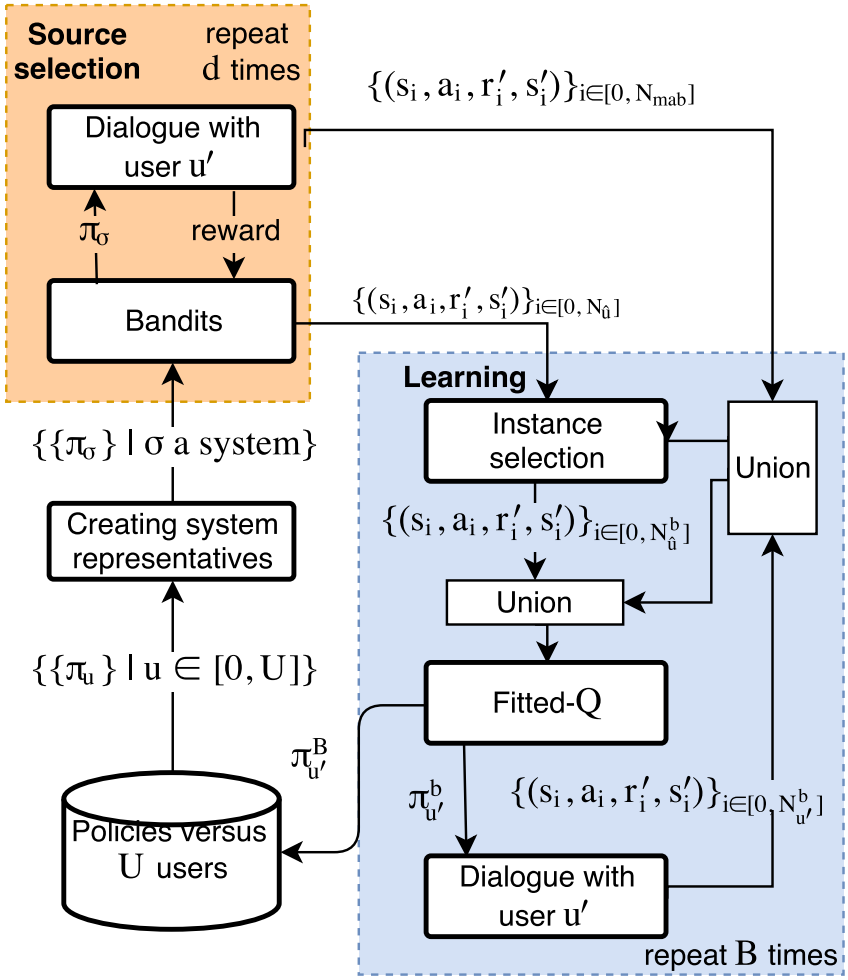


Figure 6.1: Adaptation process

Source selection

The source selection problem is cast into a MAB algorithm, implemented here as UCB1 (Auer et al. 2002b), each arm standing for a representative. When the MAB selects an arm, its corresponding policy π interacts with the user for one full dialogue. The MAB performs n_{mab} policy selections. N_{mab} transitions from dialogues, with target user u', are collected during this procedure. In the end of this initial MAB step, the representative policy that yielded the highest empirical reward designates the source from which to transfer. The algorithm transfers $N_{\hat{u}}$ transitions from its source $\hat{u}$ dialogues, to construct a batch of dialogues. Transitions from the trajectories of the chosen source are added to those already collected from the target as suggested by Lazaric et al. (2008).

Instance selection

Source transitions are subject to an instance selection to alleviate bias when sufficient target data has been collected. After instance selection, the $N^b_{\hat{u}}$ remaining transitions are added to the target transitions for training a first policy with Fitted-Q (FTQ). The idea is to only transfer transitions that are not present in the target transition dataset. We use an algorithm called Density-Based (Genevay et al. 2016) :given a parameter η and given a transition from the source (s, a, r', s'), all the transitions from the target MDP which contain action a are considered. If there is a source transition (s_i, a, r'_i, s'_i) such that $||s - s_i||_2 \leq \eta$ then the transition is not added to the batch. The choice of η is problem-dependent and should be tuned carefully. A large value for this parameter leads to adding too few transitions to the batch, while a small value might make the source bias pertain too long.

6.2.2 The *learning* phase

The hybrid source-target dataset is used for training the current policy that controls the behaviour during the next epoch, with an ε-greedy exploration. $N^b_{u'}$ transitions are collected this way, and used to refine its training. The algorithm repeats the operation from the transition selection step for B minibatches. Note that between each minibatch, Density-Based is used to remove redondant source transitions. Eventually, the final learnt policy $\pi^B_{u'}$ on u' is added to the database. Note that this policy does not explore anymore.

6.3 Source representatives

This section presents the main contributions of the chapter. The adaptation process requires a setup of several source representatives in order to do the first dialogues, with a target user, handled by the MAB process. Indeed, setting one arm for every source policy is not sustainable for real-world systems since the stochastic MAB regret is linear in number of arms. The initial phase of MAB dialogue collection lasts $d \sim 100$ dialogues. This is the reason why in this chapter we propose to create a set of limited size k of source representatives from a large user database. Two methods are proposed: one based on the cost function of k-medoids and the other one based on k-means. Both rely on **PD-DISTANCE**, a novel policy-driven distance that we introduce in this chapter:

$$d_{pd}(u, u') = \sqrt{\sum_{s \in \Omega} 1 - \mathbb{1}\left(\pi_u(s), \pi_{u'}(s)\right)} \tag{6.1}$$

where u and u' are source users and π_u and $\pi_{u'}$ the deterministic policies trained with them. The state set Ω is obtained by sampling over the states contained in the dialogues database. The function $\mathbb{1}$ is the Kronecker delta: $\mathbb{1}(x, y) = \delta_{xy}$

In the **KMEDOIDS** method, we propose to choose directly k representatives into the systems database. The cost function optimised by the k-medoids algorithm, denoted as J here, is used. Let $\mathscr{P}_k(\mathscr{U})$ denote the ensemble of k combinations of elements among $\mathscr{U}$, the set of all source users. If $U \in \mathscr{P}_k(\mathscr{U})$, and d is a distance, then the cost function is defined as:

$$J(U) = \sum_{u \in \mathscr{U}} \min_{u' \in U} d(u, u'). \tag{6.2}$$

Thus, the goal is to find the set P_{min} minimising KMEDOIDS. This chapter uses PD-DISTANCE as the distance d. For performance reasons, instead of optimising over all $U \in \mathscr{P}_k(\mathscr{U})$, we sample uniformly on $\mathscr{P}_k(\mathscr{U})$ and keep the smallest cost value $J(U)$, but one could use better optimisation methods to find the best fit according to KMEDOIDS (like a greedy approach).

In the **KMEANS** method, we cluster systems with the k-means algorithm using PD-DISTANCE as a distance. In order to keep using the Euclidean distance in the k-means algorithm, one must design each vector v to cluster this way: $v(s, a) = 1$ if a has been chosen in s, 0 otherwise. Note that KMEDOIDS directly picks elements from the main set while k-means regroups elements around means of vectors potentially corresponding to non-existent systems. The KMEANS method must construct the k system representatives from the clusters. A representative is a new system learnt using FTQ. The training batch is constructed by gathering N_{ts} transfer transitions (s, a, r', s') of each system of the corresponding cluster.

6.4 Experiments

In order to test the previous methods, experiences are ran on the Negociation Dialogue Game (NDG) (Laroche and Genevay 2017). It is a slot-filling problem as described in Section 3.2. In this game, two players must agree on a time-slot for an appointment. For each player p, each time-slot τ is associated to a cost $c_{p,\tau} \in [0, 1]$. Each player knows its own costs, but does not know the costs associated to the other player. At each turn of the game, a player can refuse the other player's time-slot and propose another time-slot: `RefProp`(τ), ask the other player to repeat: `AskRepeat`, terminate the game: `EndDial` or accept the other player's slot: `Accept`. The noise inherent to spoken dialogues (because of Automatic Speech Recognition (ASR) errors) is simulated: when a player proposes a time-slot, there is a probability ξ that the time-slot proposed is corrupted where ξ denotes the Sentence Error Rate (SER) of this player.

The Speech Recognition Score (SRS) ν of an utterance is then computed according to the following formula (Khouzaimi et al. 2015):

$$\nu = \frac{1}{1 + e^{-x}}$$

where $x \sim \mathcal{N}(\mu, 0.2)$, $\mu = \mu_\top$ is the probability of a proper understanding,and $\mu = \mu_\perp$ is the probability of a miss-understanding i.e. $\mu = (1 - z)\mu_\top + z\mu_\perp$ where $z \sim \mathcal{B}(1, \xi)$. In

a nutshell, the SER denotes if an error appears or not, and the SRS denotes how confident the system is about the transcripted utterance; if there is an error, the confidence score will be low in expectation (with respect to $\mu = \mu_\perp$); if there are no errors, the confidence score will be hight in expectation (with respect to $\mu = \mu_\top$). These parameters are relative to each player. The further apart the normal distribution centers are, the easier it will be for the system to know if it understood the right time-slot, given the score. At the end of the game, if there is an agreement (*i.e.* there is no misunderstanding on the agreed slot $\tau = \tau_u = \tau_v$), the system v, receives a immediate reward $r_v = \omega_v - c_{v,\tau} + \varrho_v(\omega_u - c_{u,\tau})$, where u denotes the other player: the user (either real human or a user simulator). For each player $p \in \{u, v\}$, $\omega_p \in \mathbb{R}$ is the utility of reaching an agreement, and $\varrho_p \in \mathbb{R}$ is the cooperation parameter. If players, v and u, agreed on different time-slots, the following formula applies to compute v's immediate reward $r_v = -c_{v,\tau_v} + \varrho_v(-c_{u,\tau_u})$. In this context, players would better agree on the same time-slot even if it is costly for them to book this specific time slot.

The return is then $G_v = \gamma_v^t r_v$ where t is the lenght of the dialogue and $\gamma_v \in [0, 1]$ his patience. Thanks to the γ_v parameter, players are inclined to accept a time-slot in a limited time. In the following, the number of available slots denoted as $N_\tau \in \mathbb{N}^+$, is set to 4 and the maximum number of utterances in a dialogue is set to 50 (once this maximum is reached, a zero return is given). We set $\varrho_u = \varrho_v = 1$ and $\omega_u = \omega_v = 1$. An example of an NDG execution is displayed in the following listing:

```
# It is a 3 time-slots game (N_τ = 3), 2 players, u and v.
# Cooperation rates: ϱ_u = ϱ_v = 1.
# Utilities: ω_u = ω_v = 1.
# Patiences (discount factors): γ_v = γ_u = 0.9.
# Costs: c_{u,0} = 0, c_{u,1} = 0.75, c_{u,2} = 0.5, c_{v,0} = 0.75, c_{v,1} = 0, c_{v,2} = 0.5.
# ---------------------------------------------------------------
# Start of the dialogue.
Turn 0: u says RefProp(0) # There are no errors (z = 1), ν = 0.8.
Turn 1: v says RefProp(1) # There are no errors (z = 1), ν = 0.75.
Turn 2: u says RefProp(2) # There is an error (z = 0), ν = 0.3.
Turn 3: v says AskRepeat.
Turn 4: u says RefProp(2) # There are no errors (z = 1), ν = 0.9.
Turn 5: v says Accept.
# End of the dialogue (t = 6).
# ---------------------------------------------------------------
# Both players found a compromise.
# Immediate rewards: r_v = r_u = 1 − 0.5 + 1 − 0.5 = 1.
# Returns: G_u = G_v = 0.9^6 · 1 ≈ 0.53.
```

Example of an NDG execution.

We will test both KMEANS and KMEDOIDS methods for searching representatives. The objective is to show that these methods improve the dialogue quality compared to non adaptive methods. We did all the tests in the following context: a user (human-model user or handcrafted user) and a system play an NDG. A dialogue is defined as one trajectory of the game. Slot preferences for users and systems are determined randomly at the beginning of each dialogue. The collected target dialogues are used to train a DP for the new user and the baselines and the clustering methods are compared in their ability to enable fast user adaptation.

	Merwan	*Nico*	*Will*	*Alex*
`Accept`	7%	35%	24%	13%
`EndDial`	0%	0%	0%	0%
`AskRepeat`	1%	14%	10%	6%
`RefProp`(0)	88%	45%	60%	64%
`RefProp`(1)	3%	5%	6%	15%
`RefProp`(2)	0%	0%	1%	2%
`RefProp`(3)	1%	0%	0%	0%
`learn error`	5.2%	5.2%	4.9%	6.8%

Table 6.1: Rounded actions distributions of humans and learn error of their k Nearest Neighbours (kNN) model.

Before jumping to the results, next section presents the user ensemble design.

6.4.1 Users design

Experiments are split in two parts with different sets of (source and target) users: the first set is artificially handcrafted (handcrafted users), while the second one is trained on human-human trajectories (human-model users).

Handcrafted users:

To illustrate the need for user adaptation, different types of handcrafted users are defined:

- The deterministic user (**DU**) proposes its slots in decreasing order (in term of its own costs). If a slot proposed by the other user fits in its $x\%$ better slots, it accepts, otherwise it refuses and proposes its next best slot. If the other user proposes twice the same slot (in other words, he insists), **DU** terminates the dialogue. Once that **DU** proposed all its slots, it restarts with its best slots all over again.
- The random user (**RU**) accepts any slot with a probability of x, otherwise it refuses and proposes a random slot.
- The always-refprop-best user (**ARPBU**) always refuses other user's slot and proposes its best slot.
- The always-accept user (**AAU**) always accept the other user slot. If AAU begins the dialogue, it proposes its best slot.
- The stop-after-one-turn user (**SAOTU**) proposes a random slot then ends the dialogue regardless of the other user response.

Human-model users:

In order to gather dialogues from human users, a multi-human version of the negotiation game has been created. Making the humans play together avoids too fast adaptation from the humans (unlike human versus computer setup) and thus keep the experiments in a stationary environment. Both players know they are playing against another human.

The number of slots available has been set to $N_\tau = 4$ and all human users share the same parameters from the negotiation game which are $\gamma_u = 0.9$, $\omega = 1$, $\xi = 0.3$, $\mu_\top = 1$, $\mu_\bot = -1$ and $\varrho = 1$. The game is then fully cooperative. Four humans: *Alex*, *Nico*, *Merwan* and *Will* played an average of 100 dialogues each. Using human trajectories, we design human-model users. State/action couples are extracted from these trajectories.

Human-model users can do the following actions: `Accept`, `AskRepeat` and `EndDial`. They can also `RefProp`(τ) to refuse the other user slot and propose their τ^{th} best slot. For instance, the action `RefProp`(0) then means that the human-model user refuses and proposes its best slot. We find the corresponding human-model users actions with the humans actions. Table 6.1 shows the empirical distribution on the human-model users actions space for each (real) human. Although humans were neither playing against the same players, nor starting from the same initial states of the game, some behavioural differences clearly appear. *Merwan* tends to insist on his best slot while *Nico* seems more compliant. *Alex* is more versatile in the actions chosen.

Human-model users require an approximate representation, or projection, of the human state. The dialogue state representation is defined as a vector of the $2 + 3N_\tau$ following attributes: the SRS ν of the last received utterance, the costs of all slots sorted, the frequencies of all `RefProp`(i) actions done by this user during the dialogue, the frequencies of all slot propositions done by the other user (ordered by cost for this user) during the dialogue and finally the cost of the last slot proposed by the other user.

Each human is modelled with a kNN, with $k = 5$, fed with their corresponding data couples state/action. Table 6.1 also shows the training errors.

Finally, handcrafted and human-model users share the following parameter values: $\xi = 0.3$, $\mu_\top = 1$, $\mu_\perp = -1$, $\omega = 1$, $\varrho = 1$ and $\gamma_u = 0.9$.

6.4.2 Systems design

Each system is trained with the least-square FTQ algorithm. Their actions set is restricted to: `Accept`, `AskRepeat`, `EndDial`, and two `RefProp` actions: `RefPropNextBest` to refuse the other user's slot and propose the next best slot after the last slot the system proposed (once all slots have been proposed, the system loops) and `InsistCurrentBest` to propose his last proposed slot. The DST module collects three attributes, the current iteration number of the dialogue t, the SRS and the difference between the cost of the next slot the system can propose and the cost of the slot currently proposed by the user c. FTQ's feature representation is then defined with 7 attributes for each action:

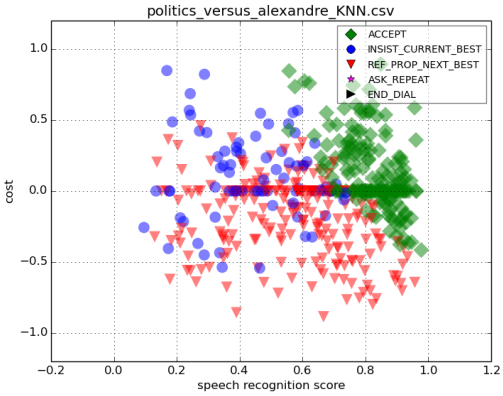
$$\phi(s, a) = (1, c, \nu, (1 - \frac{1}{t}), c \cdot \nu, c \cdot (1 - \frac{1}{t}), \nu \cdot (1 - \frac{1}{t}))$$

The learning algorithm is described as follows: given a target user u'[1], the learning is done in B minibatches of FTQ (with $\gamma = 0.9$, $\upsilon = 0.001$ and $K = 200$). For each minibatch b, a set of $N^b_{u'}$ dialogues is generated between the system and a user and then a new policy is computed with FTQ fed with all the dialogues done so far. Policies are ε-greedy, ε annealing from $\varepsilon = 0.25$ at the 1st batch to $\varepsilon = 0.01$ at the last batch. In between, $\varepsilon(b) = \frac{1}{a_e \cdot b + b_e}$ where $a_e = 19.2$, $b_e = -15.2$. ε is set to 0 during the test phase (in order to greedily exploit the current policy). In the human setup, $B = 6$ and $N^b_{u'} = 500$. In the handcrafted setup, $B = 6$ and $N^b_{u'} = 200$.

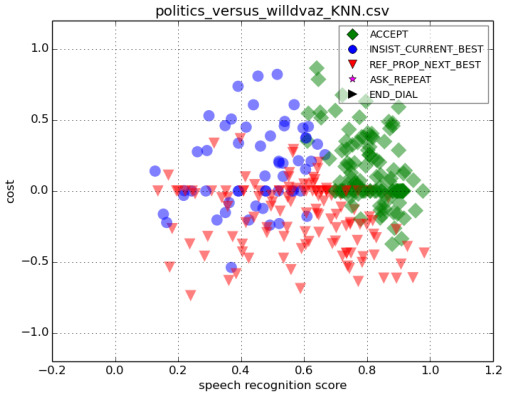
6.4.3 Cross comparisons

To show the importance of user adaptation, source systems are respectively trained versus users. Then, each system interacts with all the users and we compare the results. The

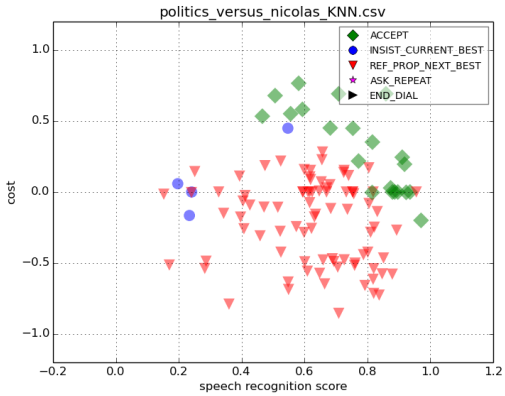
[1] We reuse notations from section 6.2.



(a) *vsAlex policy's 2D projection.*



(b) *vsWill policy's 2D projection.*



(c) *vsNico policy's 2D projection.*

Figure 6.2: Some projections of policies optimised versus human-model users

	type	$\mu_\top$	$\mu_\bot$	x	vspu1	vspu2	vspu3	vspu4	vspu5	vspu6	vspu7
pu1	DU	1	-1	0.1	**0,62**	0,44	0,46	0,40	0,40	0,40	0,59
pu2	DU	5	-5	0.1	0,53	**0,82**	0,81	0,51	0,70	0,41	0,71
pu3	DU	5	-5	0.2	0,53	**0,81**	**0,81**	0,52	0,72	0,42	0,71
pu4	RU	5	-5	0.1	0,42	0,94	0,94	**1,00**	0,92	0,85	0,94
pu5	ARPBU	1	-1		0,84	0,98	1,00	1,11	**1,16**	1,13	1,05
pu6	AAU	1	-1		0,95	1,06	1,07	1,29	1,27	**1,30**	1,06
pu7	SAOTU	1	-1		0,43	0,26	0,27	0,10	0,18	0,03	**0,58**

Table 6.2: Handcrafted users characteristics and cross comparison between handcrafted users and systems. For $p \in [0, 7]$, vspu$_p$ is the system trained versus the user pu$_p$

	vsAlex	*vsNico*	*vsWill*	*vsMerwan*
Alex	**1.077**	1.041	1.071	1.066
Nico	1.246	**1.251**	1.246	1.231
Will	1.123	1.109	**1.126**	1.117
Merwan	0.989	0.903	0.985	**0.998**

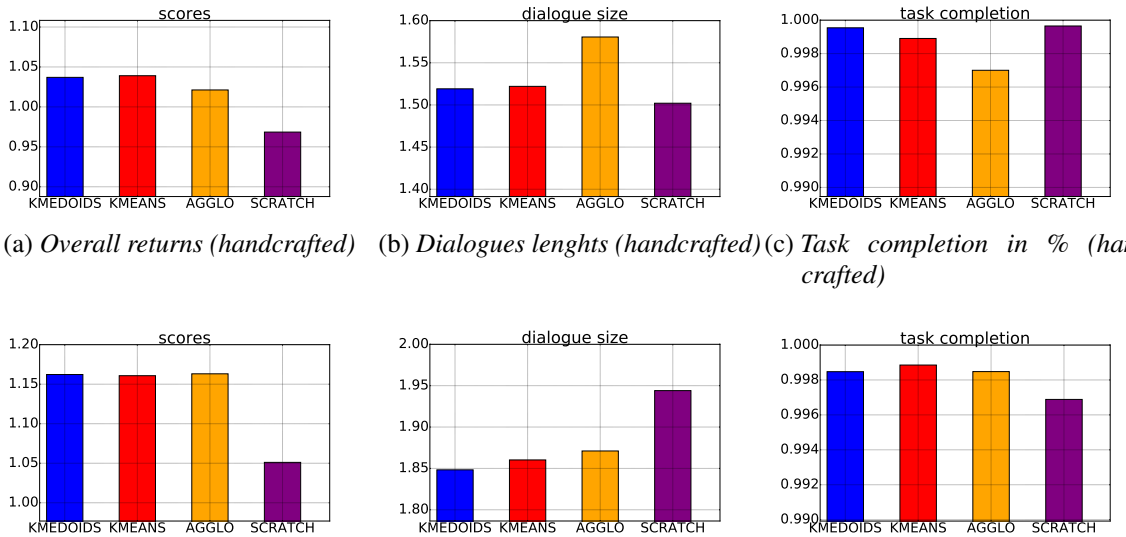
Table 6.3: Cross comparison between human-model users and systems

experiences are repeated for 10 runs. Dialogue testing size is set to 10^3 for each run. In the **handcrafted setup**, as in Genevay et al. (2016), handcrafted users are created. Parameters of these users are listed in Table 6.2. Also, cross comparisons between source users and systems are displayed. Results in bold show that each system trained versus a specific user is the best fit to dialogue with this user. One can see clear similarities between some of the results. This is where the representatives design method will operate by grouping all these similar policies.

In the **human setup**, test systems are trained against the human-model user. Results are shown in Table 6.3. Note that label *Will* means model of *Will* and not *Will* himself as well as *vsWill* means the system trained against *Will*'s model. Again, trained systems perform better than others against the user they learnt on. However, differences are not as clear as in the handcrafted setup. The reason is shown in Figure 6.2a, 6.2b and 6.2c where learnt policies are quite similar. Computed policies are tested on the states s_i from the set of $(s_i, a_i, r'_i, s'_i)_{i \in N}$ they learnt from. The (ν, c) projection explains better policy differences. One can see that *vsAlex* and *vsWill* are pretty similar as they insist often when the cost is negative, in contrary of other policies. On the other hand, *vsNico* tends to `RefProp` instead of `Accept` when the SRS is high. It is straightforward to remark that this is because *Nico* has tendencies to `Accept` more than others as we saw in Table 6.1. One can notice that even if statistics gathered from human actions distribution shows significant differences (in Table 6.1), computed policies are not necessarily different (like *vsAlex* and *vsWill*).

6.4.4 Adaptation results

Now that specialised systems have been shown to lead to better results, we test the full adaptation process with KMEDOIDS and KMEANS methods. As previously, we performed



(a) *Overall returns (handcrafted)* (b) *Dialogues lenghts (handcrafted)* (c) *Task completion in % (handcrafted)*

(d) *Overall returns (human)* (e) *Dialogues lenghts (human)* (f) *Task completion in % (human)*

Figure 6.3: Dialogue quality in the handcrafted and human setup

tests on both handcrafted and human-model users. But first, the database of source systems is constructed. We created 100 source handcrafted users and 100 source human-model users. Those are designed by changing some parameters of the vanilla users. For example, a model from Alex is changed switching its SER (ξ) from 0.3 to 0.5. Parameters take random value between the following intervals: $\mu_\top \in [0, 5]$ with $\mu_\perp = -\mu_\top$, $\xi \in [0, 0.5]$, $\varrho \in [0, 1]$, and $x \in [0.1, 0.9]$. It is useful for human setup because we do not have enough dialogue corpora to design 100 systems specialized versus 100 unique human-model users. The same method is applied to generate a large number of handcrafted users as well. For each user, a source policy is trained after 6 batches of 200 dialogues (for a total of 1200 dialogues). Each system is added to its respective database (human-model or handcrafted). We end up with 100 source trained policies with 100 source handcrafted users and 100 source trained policies with 100 human-model source users.

KMEANS and KMEDOIDS are tested for the complete adaptation process versus a base of 500 target users generated randomly (in the same way as users have been generated to create source systems). As discussed in Section 6.2, the adaptation process implies a bandit phase: 25 dialogues are done versus the target user then the mean return is saved to be plotted. Then all the transitions (s, a, r', s') are retrieved from the source system winner of the bandit. The process actually transfers a maximum of 1200 dialogues. Transfer transitions are submitted to a filtering using Density-Based selection with η parameter picked in the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ [2]. Then, a new policy is learnt with FTQ fed with transitions from the source system and transitions from the bandit dialogues. To avoid divergence, a λ-regularisation (Tikhonov 1963; Farahmand et al. 2009) is applied to FTQ with $\lambda = 1$. Once the policy learnt, 25 additional dialogues are sampled versus the target user. After this sampling, the mean return is saved to be plotted later. The process is repeated 6 times for a total of $25 + 6 \cdot 25 + 1200$ dialogues maximum for the learning

[2] We kept only $\eta = 0.3$ as results are pretty similar with any η

and $25 + 6 \cdot 25 + 25$ dialogues for the evaluation. All systems, sources and targets, share the following parameters; $\omega = 1$, $\gamma_u = 0.9$, $\xi = 0.0$, $\mu_\top = 5$, $\mu_\perp = -5$, $\varrho = 1$ but differ in their policy. All systems are learnt with FTQ using the following parameters: $\upsilon = 0.001$, $K = 200$ and $\gamma = 0.9$. They all follow an ε-greedy policy with ε defined as in Section 6.4.2.

In order to compare the previous methods, we introduce two naive ways for user adaptation, AGGLO and SCRATCH. The first one learns a unique system to represent the whole systems database and the second one adopts a random policy during the bandit phase then follows an ε-greedy policy like other methods but without any transfer. Before running experiment, pre-processing is done for some the methods: for AGGLO, 1200 dialogues are gathered among all source systems in the database. That means 12 dialogues are collected randomly from the dialogue set of each of the 100 source systems. A policy is learnt with one batch of FTQ with $\upsilon = 10^{-6}$, γ=0.9 and K=200. This policy is used to create one unique system representative for all the database. For KMEANS, PD-DISTANCE vector representations of each system in the database are created by sampling over 20000 states (picked from source systems). Then these systems are clustered with k=5 using k-means with euclidean distance. For each cluster the previous AGGLO method is applied in order to create a cluster representative. Finally, for KMEDOIDS, a random sampling of five-element sets is ran. The J value of each set is computed and the one who minimizes this value is kept. Finally, 10 different sets of AGGLO, KMEANS and KMEDOIDS are created and tested. Results are shown in Figure 6.3. In the handcrafted setup, the overall dialogue quality of the proposed methods is significantly better than AGGLO and SCRATCH baselines. Indeed, dialogues are shorter [3], final return is higher and the task is more often completed. On the other side, returns and task completions are similar in the human setup. Still, the size of the dialogues is improved by KMEANS and KMEDOIDS offering a better dialogue experience and thus users keep using the DS.

6.5 Related work

To our knowledge, only one paper treats the subject of searching system representatives among a systems database: (Mahmud et al. 2013) has a similar adaptation process as the one presented in this chapter. It is not applied to DSs specifically. In order to choose good representatives from the policies/MDP/systems database, clustering is done using the following distance:

$$d_{V(M_i,M_j)} = max\{V_i^{\pi_i^*} - V_i^{\pi_j^*}, V_j^{\pi_j^*} - V_j^{\pi_i^*}\},$$

given two MDPs M_i and M_j, where V_k^π is the return of the policy π when executed on MDP M_k and ${\pi^*}_k$ refers to the optimal policy for MDP M_k. Thus, to compute all the systems distances two by two, one needs to sample dialogues between the source users and all the source systems of the database. In a real life dialogue applications with humans, it is not possible to do such a thing unless one creates a model of each source user.

[3] SCRATCH's dialogue size can be shorter because it use random policy on jumpstart and then ends the dialogue more often.

6.6 Conclusion

In this chapter, user adaptation has been proved to improve DSs performances when users adopt different behaviours. The chapter shows that indeed, each human adopts a different way to play the NDG although the shade is subtle. So, a system learnt versus a particular user is more efficient than other systems for this user, in the handcrafted user setup as in the human-model user setup.

User adaptation requires selecting source systems to transfer knowledge. This chapter proposed 2 methods: KMEANS and KMEDOIDS combined to a novel distance PD-DISTANCE to select representative source systems, from a large database, which are used for transferring dialogue transitions. These methods outperform generic policies in the handcrafted setup and improve dialogue quality when facing models learnt on human-human data.

6.7 Discussion

The framework is working but is heavily engineered. Each composant can be tested and validated which is a good thing. However, each of those composant can generate errors and approximations. Also the number of meta-parameters make it hard to optimise from a human perspective. Finally, the effort of implementation is cumbersome. As a consequence, we will propose solutions to condense the whole pipeline into a single procedure, by extending the famous DQN algorithm. As it is an early work in progress, we will see those propositions in an extra chapter: Appendix A.

III Safe Transfer Learning

7 The Dialogue Manager as a safe policy 69

7.1 Motivation
7.2 Budgeted Dialogue Policies
7.3 Budgeted Reinforcement Learning
7.4 A scalable Implementation
7.5 Experiments
7.6 Discussion
7.7 Conclusion

8 Transfering safe policies 89

8.1 Motivation
8.2 ε-safe
8.3 Experiment

7. The Dialogue Manager as a safe policy

In this chapter and in Chapter 8, we formulate the hypothesis that the first dialogues with a new user should be handled in a conservative way, for two reasons: to avoid user dropout; and to ensure the gathering of informative dialogues to speedup the learning. We propose to design and transfer a generic safe policy to handle the early interactions with the target user. The transfering algorithm is developped in Chapter 8, but first, we study here how to compute this policy. Therefore, we cast the dialogue problem as a Budgeted Markov Decision Process (BMDP) and propose a new Reinforcement Learning (RL) framework to learn safe policies in this context. A BMDP is an extension of an Markov Decision Process (MDP) to critical applications requiring safety constraints. It relies on a notion of risk implemented in the shape of a cost signal constrained to lie below an – adjustable – threshold. For our application, the risk is described as a user hanging up the conversation. The threshold is then the maximum hangup frequency we can afford with a user. So far, BMDPs could only be solved in the case of finite state spaces with known dynamics. However, as we seen earlier, dialogue policies take as input continuous variables as the SRS and we do not know the dynamic of a human dialoguing. As a consequence, this work extends the state-of-the-art to continuous spaces environments and unknown dynamics. We show that the solution to a BMDP is a fixed point of a novel Budgeted Bellman Optimality operator. This observation allows us to introduce natural extensions of Deep Reinforcement Learning (DRL) algorithms to address large-scale BMDPs. In addition to the DS application, we validate our approach with an autonomous driving application[1]

[1]The work in the chapter has been done in collaboration with Edouard Leurent. I (Nicolas Carrara) handled the creation and implementation of a novel algorithm (BFTQ) and the early proofs of concept on 2D worlds. I also made the experiments on dialogue systems. Edouard Leurent and I worked togetheir on the proofs and a novel exploration procedure. Finally, Edouard Leurent handled the autonomous driving experiments.

7.1 Motivation

As stated in Chapter 4, RL is a general framework for decision-making under uncertainty. Formally, we seek a policy $\pi \in \mathcal{M}(A)^{\mathcal{S}}$ that maximises in expectation the γ-discounted return of rewards $G^\pi = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$.

However, this modelling assumption comes at a price: no control is given over the spread of the performance distribution (Dann et al. 2019). In many critical real-world applications, including dialogue systems, failures may turn out very costly. This is an issue as most decision-makers would rather give away some amount of expected optimality to increase the performances in the lower-tail of the distribution. This has led to the development of several risk-averse variants where the optimisation criteria include other statistics of the performance, such as the worst-case realisation (Iyengar 2005; Nilim et al. 2005; Wiesemann et al. 2013), the variance-penalised expectation (Tamar et al. 2012; Garcia et al. 2015), the Value at Risk (Mausser et al. 2003; Luenberger 2013), or the Conditional Value at Risk (Chow, Tamar, et al. 2015; Chow, Ghavamzadeh, et al. 2018).

Reinforcement Learning also assumes that the performance can be described by a single reward function R. Conversely, real problems typically involve many aspects, some of which can be contradictory (C. Liu et al. 2014). For instance, a self-driving car needs to balance between progressing quickly on the road and avoiding collisions. When aggregating several objectives in a single scalar signal, as often in Multi-Objectives Reinforcement Learning (Roijers et al. 2013), no control is given over their relative ratios, as high rewards can compensate high penalties. For instance, if a weighted sum is used to balance velocity v and crashes c, then for any given choice of weights ω the optimality equation $\omega_v \mathbb{E}[\sum \gamma^t v_t] + \omega_a \mathbb{E}[\sum \gamma^t c_t] = G^* = \max_\pi G^\pi$ is the equation of a line in $(\mathbb{E}[\sum \gamma^t v_t], \mathbb{E}[\sum \gamma^t c_t])$, and the automotive company cannot control where its optimal policy π^* lies on that line. This phenomena also occurs in dialogue systems. Let's consider a dialogue system that does not say Hello at the beginning of the dialogue; it increases the dialogue efficiency (as the task is completed faster) but impacts the dialogue quality, and may induce more users hanging up and definitive dropouts. Then, with new users, it would be preferable to say Hello while a straight to the point strategy would be more appropriate with recurrent users.

Both of these concerns can be addressed in the Constrainted Markov Decision Process (CMDP) setting (Beutler et al. 1985; Altman 1999). In this multi-objective formulation, task completion and safety are considered separately. We equip the MDP with a cost signal $C \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and a cost budget $\beta \in \mathbb{R}$. Similarly to G^π, we define the return of costs ${G_c}^\pi = \sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)$ and the new cost-constrained objective:

$$\max_{\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}} \mathbb{E}[G^\pi | s_0 = s] \quad \text{s.t.} \quad \mathbb{E}[{G_c}^\pi | s_0 = s] \leq \beta \tag{7.1}$$

This constrained framework allows for better control of the performance-safety trade-off. However, it suffers from a major limitation: the budget has to be chosen before training, and cannot be changed afterwards.

To address this concern, the BMDP was introduced in (Boutilier et al. 2016) as an extension of CMDPs to enable the online control over the budget β within an interval $\mathscr{B} \subset \mathbb{R}$ of admissible budgets. Instead of fixing the budget prior to training, the objective is now to find a generic optimal policy π^* that takes β as input so as to solve the corresponding CMDP (Eq. (7.1)) for all $\beta \in \mathscr{B}$. This gives the system designer the ability to move the

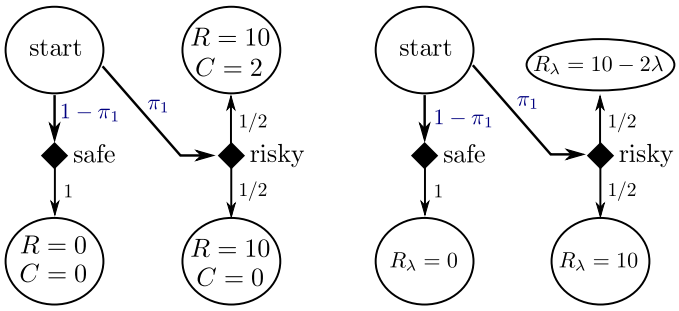


Figure 7.1: On the left hand side, a simple *risky-vs-safe* BMDP. The probability of picking the risky action is π_1. On the right hand side an attempt to relax the problem with negative rewards.

optimal policy π^* in real-time along the Pareto-optimal curve of the different reward-cost trade-offs.

7.1.1 A remark on deterministic policies

A convenient property of MDPs, is that the optimal policy is unique, deterministic and greedy: $\pi^*(s) = \arg\max_a Q(s, a)$. In a CMDP, and *a fortiori* in a BMDP, this is in general not the case. It has been shown indeed that the optimal policy under constraint is a random mixture of two deterministic policies Beutler et al. (1985, Theorem 4.4).

To illustrate this fact, let us consider the trivial BMDP on the left of Figure 7.1. On this example we have $G^\pi = 10\pi_1$ and ${G_c}^\pi = \pi_1$. The deterministic policy consisting in always picking the safe action is feasible for any $\beta \geq 0$. But if $\beta = 1/2$, the most rewarding feasible policy is to randomly pick the safe and risky actions with equal probabilities. If we attempt to cast this BMDP into an MDP by replacing the costs by negative rewards, the policy we will obtain will be deterministic, hence sub-optimal.

Our first contribution is to re-frame the original BMDP formulation in the context of continuous states and infinite discounted horizon. We then propose a novel Budgeted Bellman Optimality Operator and prove the optimal value function to be a fixed point of this operator. Second, we use this operator in BFTQ, a batch RL algorithm, for solving BMDPs Online by interacting with an environment, through function approximation and a tailored exploration procedure. Third, we scale this algorithm to large problems by providing an efficient implementation of the Budgeted Bellman Optimality operator based on convex programming, and by leveraging tools from DRL such as NNs and synchronous parallel computing. Finally, we validate our approach in two environments that display a clear trade-off between rewards and costs: a SDS and a problem of behaviour planning for autonomous driving. The proofs of our main results are provided Appendix .1.

7.2 Budgeted Dialogue Policies

We work in the space of budgeted policies, where π both depends on β and also outputs a next budget β_a. Hence, the budget β is neither fixed nor constant as in the CMDP setting but instead evolves as part of the dynamics.

We cast the BMDP problem as a multi-objective MDP problem (Roijers et al. 2013) by considering *augmented* state and action spaces $\overline{\mathcal{S}} = \mathcal{S} \times \mathscr{B}$ and $\overline{\mathcal{A}} = \mathcal{A} \times \mathscr{B}$, and equip

them with the augmented dynamics $\overline{P} \in \mathcal{M}(\overline{\mathcal{S}})^{\overline{\mathcal{S}} \times \overline{\mathcal{A}}}$ defined as:

$$\overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) = \overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \stackrel{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a), \tag{7.2}$$

where δ is the Dirac indicator distribution.

In other words, in these augmented dynamics, the output budget β_a returned at time t by a budgeted policy $\overline{\pi} \in \overline{\Pi} = \mathcal{M}(\overline{\mathcal{A}})^{\overline{\mathcal{S}}}$ will be used to condition the policy at the next timestep $t + 1$.

We stack the rewards and cost functions in a single *vectorial* signal $\overline{R} \in (\mathbb{R}^2)^{\overline{\mathcal{S}} \times \overline{\mathcal{A}}}$:

Definition 7.2.1 Given an augmented transition $(\overline{s}, \overline{a}) = ((s, \beta), (a, \beta_a))$, we define:

$$\overline{R}(\overline{s}, \overline{a}) \stackrel{\text{def}}{=} \begin{bmatrix} R(s, a) \\ C(s, a) \end{bmatrix} \in \mathbb{R}^2. \tag{7.3}$$

Likewise, we augment the return:

Definition 7.2.2 The return $\overline{G^{\pi}} = (G^{\overline{\pi}}, {G_c}^{\overline{\pi}})$ of a budgeted policy $\overline{\pi} \in \overline{\Pi}$ refers to:

$$\overline{G^{\pi}} \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} \gamma^t \overline{R}(\overline{s}_t, \overline{a}_t). \tag{7.4}$$

We also augment the value functions:

Definition 7.2.3 The value functions $\overline{V^{\pi}}$, $\overline{Q^{\pi}}$ of a budgeted policy $\overline{\pi} \in \overline{\Pi}$ are defined as:

$$\overline{V^{\pi}}(\overline{s}) = ({V_r}^{\overline{\pi}}, {V_c}^{\overline{\pi}}) \stackrel{\text{def}}{=} \mathbb{E}\left[\overline{G^{\pi}} \;\middle|\; \overline{s_0} = \overline{s}\right] \overline{Q^{\pi}}(\overline{s}, \overline{a}) = ({Q_r}^{\overline{\pi}}, {Q_c}^{\overline{\pi}}) \stackrel{\text{def}}{=} \mathbb{E}\left[\overline{G^{\pi}} \;\middle|\; \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right]. \tag{7.5}$$

We restrict $\overline{\mathcal{S}}$ to feasible budgets only: $\overline{\mathcal{S}}_f \stackrel{\text{def}}{=} \{(s, \beta) \in \overline{\mathcal{S}} : \exists \overline{\pi} \in \overline{\Pi}, {V_c}^{\overline{\pi}}(s) \leq \beta\}$ that we assume is non-empty for the BMDP to admit a solution. We still write $\overline{\mathcal{S}}$ in place of $\overline{\mathcal{S}}_f$ for brevity of notations.

Proposition 7.2.1 — Budgeted Bellman Evaluation. The value functions $\overline{V^{\pi}}$ and $\overline{Q^{\pi}}$ verify:

$$\overline{V^{\pi}}(\overline{s}) = \sum_{\overline{a} \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}|\overline{s}) \overline{Q^{\pi}}(\overline{s}, \overline{a}) \qquad \overline{Q^{\pi}}(\overline{s}, \overline{a}) = \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) \overline{V^{\pi}}(\overline{s}'). \tag{7.6}$$

Moreover, consider the Budgeted Bellman Evaluation operator $\overline{\mathcal{T}^{\pi}}$: $\forall \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}}, \overline{s} \in \overline{\mathcal{S}}, \overline{a} \in \overline{\mathcal{A}}$,

$$\overline{\mathcal{T}^{\pi}}\overline{Q}(\overline{s}, \overline{a}) \stackrel{\text{def}}{=} \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{P}(\overline{s}'|\overline{s}, \overline{a}) \overline{\pi}(\overline{a}'|\overline{s}') \overline{Q}(\overline{s}', \overline{a}'). \tag{7.7}$$

Then $\overline{\mathcal{T}^{\pi}}$ is a γ-contraction and $\overline{Q^{\pi}}$ is its unique fixed point.

Proof. The proof is available in Appendix .1.1 ■

Definition 7.2.4 — Budgeted Optimality. We now come to the definition of budgeted optimality. We want an optimal budgeted policy to: (i) respect the cost budget β, (ii) maximise the γ-discounted return of rewards G, (iii) in case of tie, minimise the γ-discounted return of costs G_c. To that end, we define for all $\overline{s} \in \overline{\mathcal{S}}$:

(i) Admissible policies $\overline{\Pi}_a$:

$$\overline{\Pi}_a(\overline{s}) \stackrel{\text{def}}{=} \{\overline{\pi} \in \overline{\Pi} : V_c^{\overline{\pi}}(\overline{s}) \leq \beta\} \text{ where } \overline{s} = (s, \beta) \tag{7.8}$$

(ii) Optimal value function for rewards V_r^* and candidate policies $\overline{\Pi}_r$:

$$V_r^*(\overline{s}) \stackrel{\text{def}}{=} \max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} V_r^{\overline{\pi}}(\overline{s}) \qquad \overline{\Pi}_r(\overline{s}) \stackrel{\text{def}}{=} \arg\max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} V_r^{\overline{\pi}}(\overline{s}) \tag{7.9}$$

(iii) Optimal value function for costs V_c^* and optimal policies $\overline{\Pi}^*$:

$$V_c^*(\overline{s}) \stackrel{\text{def}}{=} \min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} V_c^{\overline{\pi}}(\overline{s}), \qquad \overline{\Pi}^*(\overline{s}) \stackrel{\text{def}}{=} \arg\min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} V_c^{\overline{\pi}}(\overline{s}) \tag{7.10}$$

We define the budgeted action-value function $\overline{Q}^*$ similarly:

$$Q_r^*(\overline{s}, \overline{a}) \stackrel{\text{def}}{=} \max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} Q_r^{\overline{\pi}}(\overline{s}, \overline{a}) \qquad Q_c^*(\overline{s}, \overline{a}) \stackrel{\text{def}}{=} \min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} Q_c^{\overline{\pi}}(\overline{s}, \overline{a}) \tag{7.11}$$

and denote $\overline{V}^* = (V_r^*, V_c^*)$, $\overline{Q}^* = (Q_r^*, Q_c^*)$.

Theorem 7.2.2 — Budgeted Bellman Optimality. The optimal budgeted action-value function $\overline{Q}^*$ verifies:

$$\overline{Q}^*(\overline{s}, \overline{a}) = \overline{\mathcal{T}}^* \overline{Q}^*(\overline{s}, \overline{a}) \stackrel{\text{def}}{=} \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}(\overline{s}' | \overline{s}, \overline{a}) \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{\pi}_{\text{greedy}}(\overline{a}' | \overline{s}'; \overline{Q}^*) \overline{Q}^*(\overline{s}', \overline{a}'), \tag{7.12}$$

where the greedy policy $\overline{\pi}_{\text{greedy}}$ is defined by: $\forall \overline{s} = (s, \beta) \in \overline{\mathcal{S}}, \overline{a} \in \overline{\mathcal{A}}, \forall \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{A}} \times \overline{\mathcal{S}}}$,

$$\overline{\pi}_{\text{greedy}}(\overline{a} | \overline{s}; \overline{Q}) \in \arg\min_{\rho \in \overline{\Pi}_r^{\overline{Q}}} \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_c(\overline{s}, \overline{a}), \tag{7.13a}$$

$$\text{where} \quad \overline{\Pi}_r^{\overline{Q}} \stackrel{\text{def}}{=} \arg\max_{\rho \in \mathcal{M}(\overline{\mathcal{A}})} \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_r(\overline{s}, \overline{a}) \tag{7.13b}$$

$$\text{s.t.} \quad \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_c(\overline{s}, \overline{a}) \leq \beta. \tag{7.13c}$$

Proof. The proof is available in Appendix .1.2 ■

R [Appearance of the greedy policy] In classical RL, the greedy policy takes a simple form $\pi_{\text{greedy}}(s; Q^*) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$, and the term $\pi_{\text{greedy}}(a'|s'; Q^*)Q^*(s', a')$ in (7.12) conveniently simplifies to $\max_{a' \in \mathcal{A}} Q^*(s', a')$. Unfortunately, in a budgeted setting the greedy policy requires solving the nested constrained optimisation program (7.13) at each state and budget in order to apply this Budgeted Bellman Optimality operator.

Proposition 7.2.3 — Optimality of the greedy policy. The greedy policy $\overline{\pi}_{\text{greedy}}(\cdot\,; \overline{Q}^*)$ is *uniformly* optimal: for all $\overline{s} \in \overline{\mathcal{S}}$, $\overline{\pi}_{\text{greedy}}(\cdot\,; \overline{Q}^*) \in \overline{\Pi}^*(\overline{s})$. In particular, $\overline{V}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)} = \overline{V}^*$ and $\overline{Q}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)} = \overline{Q}^*$.

Proof. The proof is available in Appendix .1.3 ■

Theorem 7.2.4 — Contractivity of $\overline{\mathcal{T}}^*$. For any BMDP $(\mathcal{S}, \mathcal{A}, P, R, C, \gamma)$ with $|\mathcal{A}| \geq 2$, $\overline{\mathcal{T}}^*$ is not a contraction.

Proof. The proof is available in Appendix .1.4 ■

Budgeted Value Iteration

The Budgeted Bellman Optimality equation is a fixed-point equation, which motivates the introduction of a fixed-point iteration procedure. We introduce Algorithm 3, a Dynamic Programming algorithm for solving known BMDPs.

Algorithm 3: Budgeted Value-Iteration

Data: P, R, C
Result: $\overline{Q}^*$
$\overline{Q}_0 \leftarrow 0$
repeat
  $\overline{Q}_{k+1} \leftarrow \overline{\mathcal{T}}^* \overline{Q}_k$
until *convergence*

Unfortunately, as $\overline{\mathcal{T}}^*$ is not a contraction, we can guarantee neither the convergence of this procedure nor the unicity of its fixed points. Despite those theoretical limitations, we empirically observed the convergence to a fixed point in our experiments (Section 7.5). We conjecture a possible explanation:

Theorem 7.2.5 — Contractivity of $\overline{\mathcal{T}}^*$ on smooth $\overline{Q}$-functions. The operator $\overline{\mathcal{T}}^*$ is a contraction when restricted to the subset $\mathcal{L}_\gamma$ of $\overline{Q}$-functions such that "Q_r is L-Lipschitz with respect to Q_c", with $L < \frac{1}{\gamma} - 1$.

Proof. The proof is available in Appendix .1.5 ■

7.3 Budgeted Reinforcement Learning

In this section, we consider BMDPs with unknown parameters that must be solved by interaction with an environment.

7.3.1 Budgeted Fitted-Q

When the BMDP is unknown, we need to adapt Algorithm 3 to work with a batch of transitions $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, \overline{r}_i, \overline{s}'_i\}_{i \in [0,N]}$ collected by interaction with the environment. Applying $\overline{\mathcal{T}}^*$ in (7.12) would require computing an expectation $\mathbb{E}_{\overline{s}' \sim \overline{P}}$ over next states $\overline{s}'$ and hence an access to the model $\overline{P}$. We instead use $\hat{\overline{\mathcal{T}}}^*$, a sampling operator, in which this expectation is replaced by:

$$\hat{\overline{\mathcal{T}}}^* \overline{Q}(\overline{s}_i, \overline{a}_i, \overline{r}_i, \overline{s}'_i) \stackrel{\text{def}}{=} \overline{r}_i + \gamma \sum_{\overline{a}'_i \in \overline{\mathcal{A}}} \overline{\pi}_{\text{greedy}}(\overline{a}'_i | \overline{s}'_i; \overline{Q}) \overline{Q}(\overline{s}'_i, \overline{a}'_i).$$

We introduce in Algorithm 4 the BFTQ algorithm, an extension of the FTQ algorithm adapted to solve unknown BMDPs. Because we work with continuous state space $\mathcal{S}$ and budget space $\mathscr{B}$, we need to employ function-approximation in order to generalise to nearby states and budgets. Following the same idea used in FVI, we introduce a projection operator $\overline{\Xi}$ onto $\overline{\mathcal{F}}$, the set of representable functions with domain $\overline{\mathcal{S}} \times \overline{\mathcal{A}}$ into $\mathbb{R}^2$: $\overline{\Xi} f = \arg\min_{\widetilde{f} \in \overline{\mathcal{F}}} ||f - \widetilde{f}||_\infty$. BFTQ simply iterates over the following equation: $\overline{Q}_{k+1} \leftarrow \overline{\Xi} \hat{\overline{\mathcal{T}}}^* \overline{Q}_k$.

To operate the projection we choose a least-square regression algorithm. Precisely, given a parametrised model $\overline{Q}_\theta$, we seek to minimise a regression loss $\mathcal{L}(\overline{Q}_\theta, \overline{Q}_{\text{target}}; \mathcal{D}) = \sum_{\mathcal{D}} ||\overline{Q}_\theta(\overline{s}, \overline{a}) - \overline{Q}_{\text{target}}(\overline{s}, \overline{a}, \overline{r}, \overline{s}')||_2^2$. Any model can be used, such as linear models, regression trees, or neural networks.

Algorithm 4: Budgeted Fitted-Q

Data: $\mathcal{D}$
Result: $\overline{Q}^*$
$\overline{Q}_0 \leftarrow 0$
repeat
  $\theta_{k+1} \leftarrow \arg\min_\theta \mathcal{L}(\overline{Q}_\theta, \hat{\overline{\mathcal{T}}}^* \overline{Q}_{\theta_k}; \mathcal{D})$
until *convergence*

7.3.2 Risk-sensitive exploration

In order to run Algorithm 4, we must first gather a batch of transitions $\mathcal{D}$. Ideally we would need transitions from the asymptotic state-budget distribution $\lim_{t \to \infty} \mathbb{P}(\overline{s}_t)$ induced by an optimal policy $\overline{\pi}^*$ given an initial distribution $\mathbb{P}(\overline{s}_0)$, but as we are actually building this policy, it is not possible. Following the same idea of ε-greedy exploration for FTQ, we introduce an algorithm for risk-sensitive exploration. We follow an exploration policy: a mixture between a random budgeted policy $\overline{\pi}_{\text{rand}}$ and the current greedy policy $\overline{\pi}_{\text{greedy}}$. The batch $\mathcal{D}$ is split into several minibatches generated sequentially, and $\overline{\pi}_{\text{greedy}}$ is updated by running Algorithm 4 on $\mathcal{D}$ upon mini-batch completion. $\overline{\pi}_{\text{rand}}$ is designed to obtain trajectories that only explore feasible budgets: we impose that the joint distribution $\mathbb{P}(a, \beta_a | s, \beta)$ verifies $\mathbb{E}[\beta_a] \leq \beta$. This condition defines a probability simplex $\Delta_{\overline{\mathcal{A}}}$ from

which we transition uniformly. Finally, when interacting with an environment the initial state s_0 is usually sampled from a starting distribution $\mathbb{P}(s_0)$. In the budgeted setting, we also need to transition the initial budget β_0. Importantly, we pick a uniform distribution $\mathbb{P}(\beta_0) = \mathcal{U}(\mathscr{B})$ so that the entire range of risk-level is explored, and not only reward-seeking behaviours as would be the case with a traditional risk-neutral ε-greedy strategy. The pseudo-code of our exploration procedure is shown in Algorithm 5.

Algorithm 5: Risk-sensitive exploration

Data: An environment, a BFTQ solver, W CPU workers
Result: A batch of transitions $\mathcal{D}$

```
D ← {}
for each intermediate batch do
    split trajectories between W workers
    for each trajectory in batch do   // run this loop on each worker in parallel
        sample initial budget β ~ U(B(1, B)).
        while trajectory not done do
            update ε from schedule.
            sample z ~ U([0, 1]).
            if z < ε then sample (a, β_a) ~ U(Δ_{A×B}).          // Explore
            else sample (a, β_a) ~ π̄_greedy(a, β_a | s, β; Q̄*).   // Exploit
            append transition (s̄ = (s, β), ā = (a, β_a), r̄ = (r, c), s̄' = (s', β_a)) to batch D.
            step trajectory budget β ← β_a
        end
    end
    π̄_greedy(· ~; Q̄*) ← BFTQ(D).
end
return the batch of transitions D
```

7.4 A scalable Implementation

In this section, we introduce an implementation of the BFTQ algorithm designed to operate efficiently and handle large batches of experiences $\mathcal{D}$.

7.4.1 How to compute the greedy policy?

As stated in Section 7.2, computing the greedy policy $\overline{\pi}_{\text{greedy}}$ in (7.12) is not trivial since it requires solving the nested constrained optimisation program (7.13).

However, it can be solved efficiently by exploiting the *structure* of the set of solutions with respect to β, that is, concave and increasing.

Proposition 7.4.1 — Equality of $\overline{\pi}_{\text{greedy}}$ and $\overline{\pi}_{\text{hull}}$. Algorithm 3 and Algorithm 4 can be run by replacing $\overline{\pi}_{\text{greedy}}$ in the equation (7.12) of $\overline{\mathcal{T}}$ with $\overline{\pi}_{\text{hull}}$ as described in Algorithm 6.

$$\overline{\pi}_{\text{greedy}}(\overline{a}|\overline{s}; \overline{Q}) = \overline{\pi}_{\text{hull}}(\overline{a}|\overline{s}; \overline{Q})$$

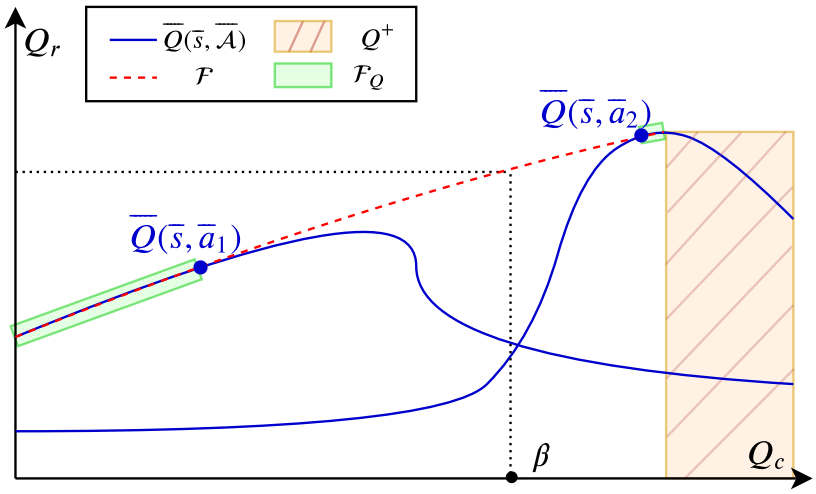


Figure 7.2: Representation of $\overline{\pi}_{\text{hull}}$. When the budget lies between $\overline{Q}(\overline{s}, \overline{a}_1)$ and $\overline{Q}(\overline{s}, \overline{a}_2)$, two points of the top frontier of the convex hull, then the policy is a mixture of these two points.

Proof. The proof is available in Appendix .1.6 ■

Algorithm 6: Convex hull policy $\overline{\pi}_{\text{hull}}(\overline{a}|\overline{s}; \overline{Q})$

Data: $\overline{s} = (s, \beta)$, $\overline{Q}$

$\overline{Q}^+ \leftarrow \{Q_c > \min\{Q_c(\overline{s}, \overline{a}) \text{ s.t. } \overline{a} \in \arg\max_{\overline{a}} Q_r(\overline{s}, \overline{a})\}\}$ `// dominated points`

$\mathcal{F} \leftarrow$ top frontier of `convex_hull`$(\overline{Q}(\overline{s}, \overline{\mathcal{A}}) \setminus \overline{Q}^+)$ `// candidate mixtures`

$\mathcal{F}_{\overline{Q}} \leftarrow \mathcal{F} \cap \overline{Q}(\overline{s}, \overline{\mathcal{A}})$

for *points* $q = \overline{Q}(\overline{s}, \overline{a}) \in \mathcal{F}_{\overline{Q}}$ *in clockwise order* **do**

  if *find two successive points* $((q_c^1, q_r^1), (q_c^2, q_r^2))$ *of* $\mathcal{F}_{\overline{Q}}$ *such that* $q_c^1 \leq \beta < q_c^2$ **then**

    $p \leftarrow (\beta - q_c^1)/(q_c^2 - q_c^1)$

    return the mixture $(1 - p)\delta(\overline{a} - \overline{a}^1) + p\delta(\overline{a} - \overline{a}^2)$

end

else return $\delta(\overline{a} - \arg\max_{\overline{a}} Q_r(\overline{s}, \overline{a}))$ `// Budget` β `always respected`

The computation of $\overline{\pi}_{\text{hull}}$ in Algorithm 6 is illustrated in Figure 7.2.

7.4.2 Function approximation

Neural Networks are well suited to model $\overline{Q}$-functions in RL algorithms (Riedmiller 2005; Mnih et al. 2015). We approximate $\overline{Q} = (Q_r, Q_c)$ using one single NN. Thus, the two components are jointly optimised which accelerates convergence and fosters learning of useful shared representations. Moreover, as in (Mnih et al. 2015) we are dealing with a finite (categorical) action space $\mathcal{A}$, instead of including the action in the input we add the output of the $\overline{Q}$-function for each action to the last layer. Again, it provides a faster convergence toward useful shared representations and it only requires one forward pass to

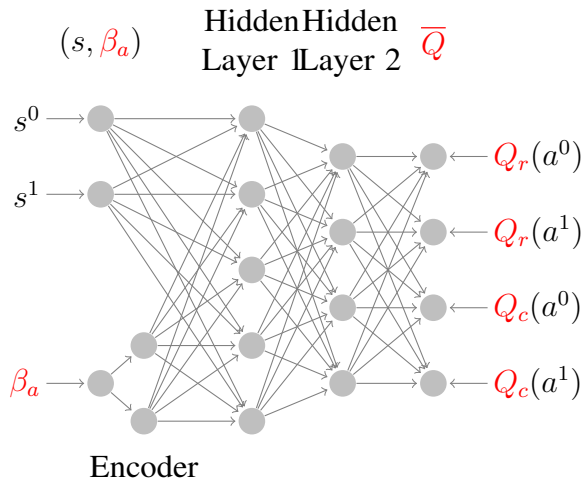


Figure 7.3: Neural Network for $\overline{Q}$-functions approximation when $\mathcal{S} = \mathbb{R}^2$ and $|\mathcal{A}| = 2$. Note that the current budget β is not an input of the network as it has no influence on the returns.

evaluate all action values. Finally, beside the state s there is one more input to a budgeted $\overline{Q}$-function: the budget β_a. This budget is a scalar value whereas the state s is a vector of potentially large size. To avoid a weak influence of the budget β_a compared to the state s in the prediction, we include an additional encoder for the budget, whose width and depth may depend on the application. A straightforward choice is a single layer with the same width as the state. The overall architecture is shown in Figure 7.3.

7.4.3 Parallel computing

In a simulated environment, a first process that can be distributed is the collection of transitions in the exploration procedure of Algorithm 5, as $\overline{\pi}_{\text{greedy}}$ stays constant within each minibatch which avoids the need of synchronisation between workers. Second, the main bottleneck of BFTQ is the computation of the target $\overline{\mathcal{T}Q}$. Indeed, when computing $\overline{\pi}_{\text{hull}}$ we must perform at each epoch a Graham-scan of complexity $\mathcal{O}(|\mathcal{A}||\widetilde{\mathscr{B}}| \log |\mathcal{A}||\widetilde{\mathscr{B}}|)$ per transition in $\mathcal{D}$ to compute the convex hulls of $\overline{Q}$ (where $\widetilde{\mathscr{B}}$ is a finite discretisation of $\mathscr{B}$). The resulting total time-complexity is $\mathcal{O}(\frac{|\mathcal{D}||\mathcal{A}||\widetilde{\mathscr{B}}|}{1-\gamma} \log |\mathcal{A}||\widetilde{\mathscr{B}}|)$. This operation can easily be distributed over several CPUs provided that we first evaluate the model $\overline{Q}(\overline{s}', \mathcal{A} \times \widetilde{\mathscr{B}})$ for each state $\overline{s}$ extracted from the dataset $\mathcal{D}$, which can be done in a single forward pass. By using multiprocessing in the computations of $\overline{\pi}_{\text{hull}}$, we enjoy a linear speedup. The full description of our scalable implementation of BFTQ is recalled in Algorithm 7.

7.5 Experiments

There are two hypotheses we want to validate.

Exploration strategies

We claimed in Section 7.3.2 that a risk-sensitive exploration was required in the setting of BMDPs. We test this hypotheses by confronting our strategy to a classical risk-neutral strategy. The latter is chosen to be a ε-greedy policy slowly transitioning from a random

Algorithm 7: Scalable Budgeted Fitted-Q

Data: $\mathcal{D}$, $\widetilde{\mathscr{B}}$ a finite subset of $\mathscr{B}$, γ, a model $\overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}} \times \overline{\mathcal{A}}}$, a regression algorithm `fit`, a set of CPU workers W

Result: $\overline{Q}^*$

$\overline{Q} \leftarrow 0$
$\overline{X} \leftarrow \{\overline{s}_i, \overline{a}_i)\}_{i \in [0, |\mathcal{D}|]}$
$\overline{S}' \leftarrow \{\overline{s}'_i\}_{i \in [0, |\mathcal{D}|]}$
repeat
  Evaluate $\overline{Q}(\overline{S}', \mathcal{A} \times \widetilde{\mathscr{B}})$ in a single forward pass
  Split $\mathcal{D}$ among workers: $\mathcal{D} = \cup_{w \in W} \mathcal{D}_w$
  for $w \in W$ **do** `// Run in parallel`
    for $(\overline{s}_i = (\cdot, \beta_i), \overline{a}_i = (\cdot, \beta_{a_i}), \overline{r}_i = (r_i, c_i), \overline{s}'_i = (s'_i, \cdot) \in \mathcal{D}$ **do**
      $\mathcal{P} \leftarrow \{(Q_c(\overline{s}'_i, \mathcal{A} \times \widetilde{\mathscr{B}}), Q_r(\overline{s}'_i, \mathcal{A} \times \widetilde{\mathscr{B}}))\}$
      $\mathcal{P}$.`prune()` `// Remove all dominated points`
      $\mathcal{H} \leftarrow$ `convex_hull`$(\mathcal{P})$.vertices() `// in cw order`
      $k \leftarrow \min\{k : \beta_i \geq q_c \text{ with } (q_c, q_r) = \mathcal{H}[k]\}$
      $q_c^2, q_r^2, q_c^1, q_r^1 \leftarrow \mathcal{H}[k], \mathcal{H}[k-1]$
      $p \leftarrow (\beta_{a_i} - q_a^1)/(q_c^2 - q_c^1)$
      $Y_c^{w,i} \leftarrow c_i + \gamma((1-p)q_c^1 + pq_c^2)$
      $Y_r^{w,i} \leftarrow r_i + \gamma((1-p)q_r^1 + pq_r^2)$
    end
  end
  Join the results: $\overline{Y} \leftarrow \cup_{w \in W}(Y_c^w, Y_r^w)$
  $\overline{Q} \leftarrow$ `fit`$(\overline{X}, \overline{Y})$
until *convergence*

to a greedy policy[2] that aims to maximise $\mathbb{E}_\pi G^\pi$ regardless of $\mathbb{E}_\pi {G_c}^\pi$. The quality of the resulting batches $\mathcal{D}$ is assessed by training a BFTQ policy and comparing the resulting performance.

Budgeted algorithms

We compare our scalable BFTQ algorithm described in Section 7.4 to an FTQ(λ) baseline. This baseline consists in approximating the BMDP by a finite set of CMDPs problems. We solve each of these CMDP using the standard technique of Lagrangian Relaxation: the cost constraint is converted to a soft penalty weighted by a Lagrangian multiplier λ in a surrogate reward function: $\max_\pi \mathbb{E}_\pi[G^\pi - \lambda {G_c}^\pi]$. The resulting MDP can be solved by any RL algorithm, and we chose FTQ for being closest to BFTQ. In our experiments, a single training of BFTQ corresponds to 10 trainings of FTQ(λ) policies. Each run was repeated N_{seeds} times.

Parameters of the algorithms can be found in Appendix .2

7.5.1 Environments

We evaluate our method on three different environments involving reward-cost trade-offs.

Corridors

This simple environment is only meant to highlight clearly the specificity of exploration in a budgeted setting. It is a continuous grid-world with Gaussian perturbations, consisting in a maze composed of two corridors: a risky one with high rewards and costs, and a safe one with low rewards and no cost. In both corridors the outermost cell is the one yielding the most reward, which motivates a deep exploration.

Parameter	Description	Value
-	Size of the environment	7 x 6
-	Standard deviation of the Gaussian noise applied to actions	(0.25,0.25)
H	Trajectory duration	9

Table 7.1: Parameters of `Corridors`

Spoken Dialogue Systems

Our second application is a dialogue-based slot-filling simulation that has already benefited from batch RL optimisation in the past (L. Li, J. Williams, et al. 2009; Chandramohan, Geist, and Pietquin 2010; Pietquin et al. 2011). As described in Section 3.2, the system fills in a form of slot-values by interacting a user through speech, before sending them a response. For example, in a restaurant reservation domain, it may ask for three slots: the area of the restaurant, the price-range and the food type. The user could respectively provide those three slot-values: `Cambridge`, `Cheap` and `Indian-food`. In this application, we do not focus on how to extract such information from the user utterances, we rather focus on decision-making for filling in the form. To that end, the system can choose among

[2] We train this greedy policy using FTQ.

a set of generic actions. There are two ways of asking for a slot value: a slot value can be either be provided with an utterance, which may cause speech recognition errors with some probability, or by requiring the user to fill-in the slots by using a numeric pad. In this case, there are no speech recognition errors but a counterpart risk of hangup: we assume that manually filling a key-value form is time-consuming and annoying. The environment yields a reward if all slots are filled without errors, and a constraint if the user hangups. Thus, there is a clear trade-off between using utterances and potentially committing a mistake, or using the numeric pad and risking a premature hangup.

Remark on the speech recognition

As in Section 6.4, when receiving an utterance, the system can either understand it ($\mu = \mu_\top$) or misunderstand it ($\mu = \mu_\perp$) with a fixed probability called the SER ξ. Then, the SRS is simulated: $\nu = (1 + \exp(-x))^{-1}$ with $x \sim \mathcal{N}(\mu, \sigma)$. It is the confidence score of the NLU module about the last utterance. Note that here are no recognition errors ($\xi = 0$ and $\nu = 1$) when the user provides information using the numeric pad.

Parameter	Description	Value
ξ	Sentence Error Rate	0.6
$\mu_\perp$	Gaussian mean for misunderstanding	-0.25
$\mu_\top$	Gaussian mean for understanding	0.25
σ	Gaussian standard deviation	0.6
p	Probability of hangup	0.25
H	Trajectory duration	10
-	Number of slots	3

Table 7.2: Parameters of `Slot-Filling`

Autonomous driving

In our third application, we use the highway-env environment (Leurent et al. 2018) for simulated highway driving and behavioural decision-making. We define a task that displays a clear trade-off between safety and efficiency. The agent controls a vehicle with a finite set of manoeuvres implemented by low-lever controllers: $\mathcal{A}$ = {no-op, right-lane, left-lane, faster, slower}. It is driving on a two-lane road populated with other traffic participants: the vehicles in front of the agent drive slowly, and there are incoming vehicles on the opposite lane. Their behaviours are randomised, which introduces some uncertainty with respect to their possible future trajectories. The task consists in driving as fast as possible, which is modelled by a reward proportional to the velocity: $R(s_t, a_t) \propto v_t$. This motivates the agent to try and overtake its preceding vehicles by driving fast on the opposite lane. This optimal but overly aggressive behaviour can be tempered through a cost function that embodies a safety objective: $C(s_t, a_t)$ is set to $1/H$ whenever the ego-vehicle is driving on the opposite lane, where H is the trajectory horizon. Thus, the constrained signal is the maximum proportion of time that the agent is allowed to drive on the wrong side of the road.

Parameter	Description	Value
N_v	Number of vehicles	2 - 6
σ_p	Standard deviation of vehicles initial positions	100 m
σ_v	Standard deviation of vehicles initial velocities	3 m/s
H	Trajectory duration	15 s

Table 7.3: Parameters of `highway-env`

7.5.2 Results

In the following figures, each patch represents the mean and 95% confidence interval over N_{seeds} seeds of the means of $(G^{\pi}, G_c{}^{\pi})$ ($(G^{\overline{\pi}}, G_c{}^{\overline{\pi}})$ for BFTQ) over N_{trajs} trajectories. That way, we display the variation related to learning (and batches) rather than the variation in the execution of the policies.

We first bring to light the role of risk-sensitive exploration in the `corridors` environment: Figure 7.4 shows the set of trajectories collected by each exploration strategy, and the resulting performance of a budgeted policy trained on each batch. The trajectories (orange) in the risk-neutral batch are concentrated along the risky corridor (red) and ignore the safe corridor (green), which results in bad performances in the low-risk regime. Conversely, trajectories in the risk-sensitive batch (blue) are well distributed among both corridors and the corresponding budgeted policy achieves good performance across the whole spectrum of risk budgets.

In a second experiment displayed in Figure 7.5, we compare the performance of FTQ(λ) to that of BFTQ in the dialogue and autonomous driving tasks. For each algorithm, we plot the reward-cost trade-off curve. In both cases, BFTQ performs almost as well as FTQ(λ) despite only requiring a single model. All budgets are well-respected on `slot-filling`, but on `highway-env` we can observe an underestimation of Q_c, since e.g. $\mathbb{E}[G_c|\beta = 0] \simeq 0.1$. This underestimation can be a consequence of two approximations: the use of the sampling operator $\hat{\overline{\mathcal{T}}}$ instead of the true environmental operator $\overline{\mathcal{T}}$, and the use of the NN function approximation $\overline{Q}_\theta$ instead of $\overline{Q}$. Still, BFTQ provides a better control on the expected cost of the policy, than FTQ(λ). In addition, BFTQ behaves more consistently than FTQ(λ) overall, as shown by its lower extra-seed variance.

7.5.3 Budgeted Fitted-Q policy executions

In Table 7.4, we display two dialogues done with the same BFTQ policy on `slot-filling`. The policy is given two budgets to respect in expectation, $\beta = 0$ and $\beta = 0.5$. For $\beta = 0$, one can see that the system never uses the `ask_num_pad` action. Instead, it uses `ask_oral` , an action subject to recognition errors. The system keeps asking for the same slot 2, because it has the lowest SRS. It eventually summarises the form to the user, but then reaches the maximum dialogue length and thus faces a dialogue failure. For $\beta = 0.5$, the system first asks in a safe way, with `ask_oral`. It may want to `ask_num_pad` if one of the SRS is low. Then, the system proceeds to a confirmation of the slot values. If it is incorrect, the system continues the dialogue using unsafe the `ask_num_pad` action to be certain of the slot values.

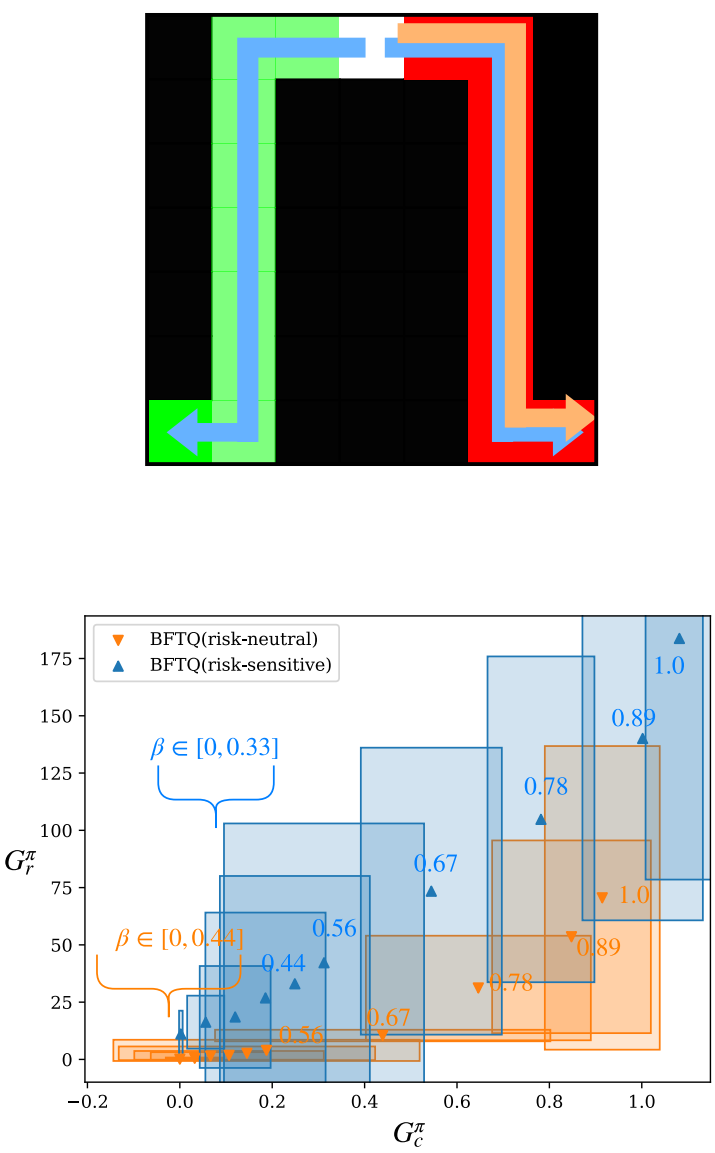


Figure 7.4: Trajectories (top) and performances (bottom) of two exploration strategies in the `corridors` environment.

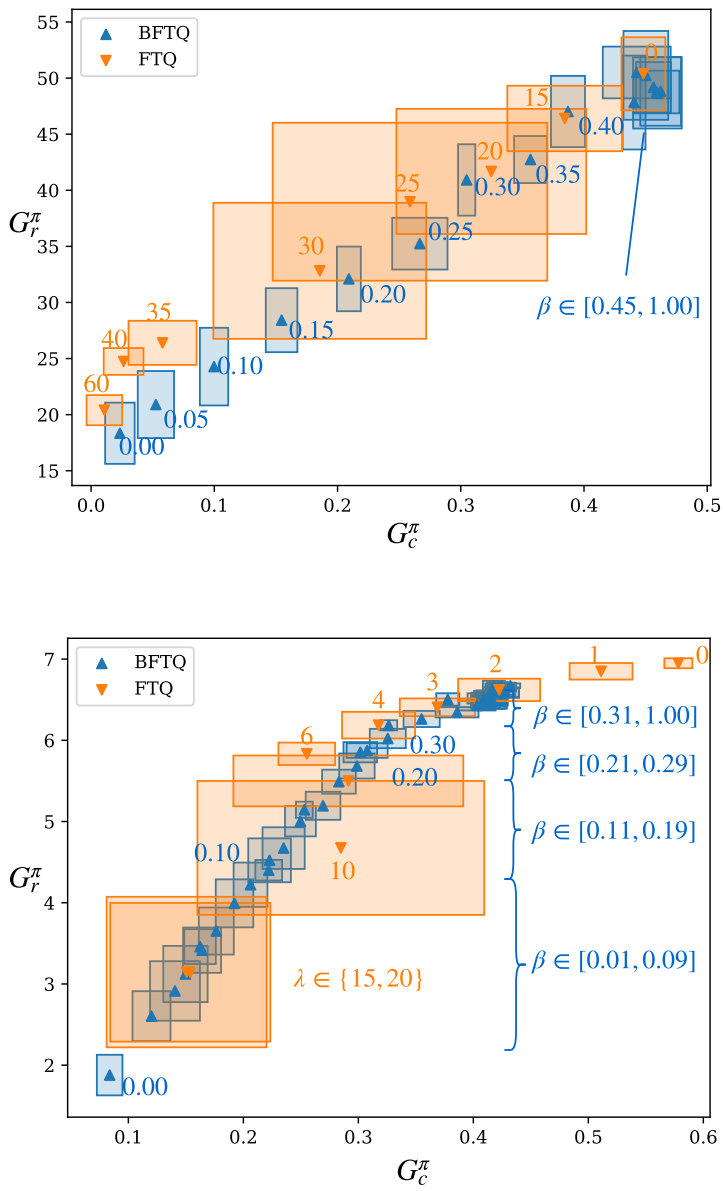


Figure 7.5: Performance comparison of FTQ(λ) and Budgeted Fitted-Q on `slot-filling` (top) and `highway-env`(bottom)

turn	$\beta = 0$	$\beta = 0.5$
turn 0	valid slots: [0, 0, 0] ν: [None None None] system says ASK_ORAL(1) user says INFORM	valid slots: [0, 0, 0] ν: [None None None] system says ASK_ORAL(2) user says INFORM
turn 1	valid slots: [0, 0, 0] ν: [None 0.48 None] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 0, 1] srs: [None None 0.56] system says ASK_ORAL(0) user says INFORM
turn 2	valid slots: [0, 0, 0] ν: [None 0.48 0.22] system says ASK_ORAL(0) user says INFORM	valid slots: [0, 0, 1] srs: [0.30 None 0.56] system says ASK_ORAL(1) user says INFORM
turn 3	valid slots: [0, 0, 0] ν: [0.62 0.48 0.22] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 0, 1] srs: [0.30 0.54 0.56] system says ASK_ORAL(0) user says INFORM
turn 4	valid slots: [0, 0, 0] ν: [0.62 0.48 0.66] system says ASK_ORAL(1) user says INFORM	valid slots: [0, 0, 1] srs: [0.68 0.54 0.56] system says ASK_NUM_PAD(1) user says INFORM
turn 5	valid slots: [0, 1, 0] ν: [0.62 0.56 0.66] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 1, 1] ν: [0.68 1.00 0.56] system says SUMMARIZE_AND_INFORM user says DENY_SUMMARIZE
turn 6	valid slots: [0, 1, 0] ν: [0.62 0.56 0.14] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 1, 1] ν: [0.68 1.00 0.56] system says ASK_NUM_PAD(2) user says INFORM
turn 7	valid slots: [0, 1, 1] ν: [0.62 0.56 0.30] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 1, 1] ν: [0.68 1.00 1.00] system says SUMMARIZE_AND_INFORM user says DENY_SUMMARIZE
turn 8	valid slots: [0, 1, 1] ν: [0.62 0.56 0.49] system says ASK_ORAL(2) user says INFORM	valid slots: [0, 1, 1] ν: [0.68 1.00 1.00] system says ASK_NUM_PAD(0) user hangs up !
turn 9	valid slots: [0, 1, 1] ν: [0.62 0.56 0.65] system says SUMMARIZE_AND_INFORM max size reached !	

Table 7.4: Two dialogues generated by a safe policy ($\beta = 0$) on the left and a risky one ($\beta = 0.5$) on the right

7.6 Discussion

Algorithm 4 is an algorithm for solving large unknown BMDPs with continuous states. To the best of our knowledge, there is no algorithm in the current literature that combines all those features.

Algorithms have been proposed for CMDPs, which are less flexible sub-problems of the more general BMDP. When the environment parameters (P, R, C) are known but not tractable, solutions relying on function approximation (Undurti et al. 2010) or approximate linear programming (Poupart et al. 2015) have been proposed. For unknown environments, Online algorithms (Geibel et al. 2005; Abe et al. 2010; Achiam et al. 2017; Chow, Ghavamzadeh, et al. 2018) and a batch algorithm (Thomas et al. 2015; Petrik et al. 2016; Laroche, Trichelair, et al. 2019; Nadjahi et al. 2019; Le et al. 2019) can solve large unknown CMDPs. Nevertheless, these approaches are limited in that the constraints thresholds are fixed prior to training and cannot be updated in real-time at policy execution to select the desired level of risk.

Budgeted Markov Decision Processes algorithms

To our knowledge, there were only two ways of solving a BMDP. The first one is to approximate it with a finite set of CMDPs (e.g. see our FTQ(λ) baseline). As explained on Figure 7.6, the optimal deterministic policy can be obtained by a line-search on the Lagrange multiplier values λ. Then, according to Beutler et al. (1985, Theorem 4.4), the optimal policy is a randomised mixture of two deterministic policies: the safest deterministic policy that violates the constraint $\pi_{\lambda-}$ and the riskier of the feasible ones $\pi_{\lambda+}$. So FTQ can be easily adapted for continuous states CMDP and BMDP through this methodology, but given the high variance it requires a lot of simulations to get a proper estimate of the calibration curve. Our solution not only requires one single model but also avoids any supplementary interaction.

The only other existing BMDP algorithm, and closest work to ours, is the DP algorithm proposed by Boutilier et al. (2016). However, their work was established for finite state spaces only, and their solution relies heavily on this property. For instance, they enumerate and sort the next states $s' \in \mathcal{S}$ by their expected value-by-cost, which could not be performed in a continuous state space $\mathcal{S}$. Moreover, they rely on the knowledge of the model (P, R, C), and do not address the question of learning from interaction data.

7.7 Conclusion

The BMDP framework is a principled framework for safe decision making under uncertainty, which could be beneficial to the diffusion of RL in industrial applications, particularly dialogue systems. However, BMDPs could so far only be solved in finite state spaces which limits their interest when dealing with continuous variable, as SRS for example. We extend their definition to continuous states by introducing a novel DP operator, that we build upon to propose a RL algorithm. In order to scale to large problems, we provide an efficient implementation that exploits the structure of the value function and leverages tools from Distributed DRL. We show that on two practical tasks, including a dialogue application, our solution performs similarly to a baseline Lagrangian relaxation method while only requiring a single model to train, and relying on an interpretable β instead of the tedious tuning of the penalty λ. As a control is given over the hanging up

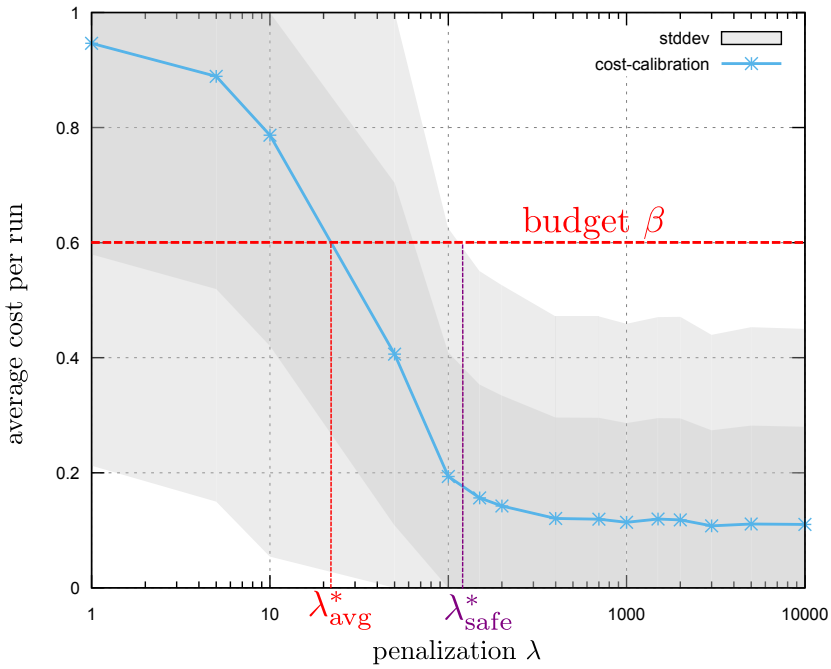


Figure 7.6: Calibration of a penalty multiplier according to the budget β. The optimal multiplier λ^*_{avg} is the smallest one to satisfy the budget constraint on average. Safer policies can also be selected according to the largest deviation from this mean cost.

frequency of an user, this framework makes a good candidate to design transferable safe policies for DS and this is the idea of the next chapter[3].

[3]Please note that we will use lagragian relaxation policies first to show if the idea of transferring safe policies actually works. As it won't work in our setting, we won't need to use BMDP policies.

8. Transfering safe policies

In this chapter, we propose to transfer a safe strategy to initiate the first dialogues. We introduce an extension of the classic ε-greedy exploration strategy. We test the algorithm on a slot-filling application.

8.1 Motivation

During its early steps of learning, an RL based dialogue agent does a lot of exploration that may lead to penalising behaviours. However, the first interactions between a user and a DS are crucial to gain trust. To improve jumpstart performance of an RL agent, one can transfer a strategy (Taylor and Stone 2009; Lazaric 2012). In dialogue, the strategy focuses on the success of the dialogue while minimising its length (Chandramohan, Geist, and Pietquin 2010; Casanueva et al. 2015b; Genevay et al. 2016). Rushing the dialogue may be problematic with some users and induce premature dialogue hangups. First, it could lead to the loss of this user once for all. Second, the lack of sucessful dialogues may affect the learning speed of the RL agent. To this extend, we introduce a novel algorithm: ε-safe. It transfers a safe strategy which avoid any critical dialogue act to prevent the aforementioned problems.

8.2 ε-safe

ε-safe (Figure 8.1) is a Q-learning algorithm (J. C. H. Watkins et al. 1992) where each action is decided by a randomly chosen policy among the greedy policy, an exploratory policy and the transferred safe policy. The safe policy may be a handcrafted or a trained policy.

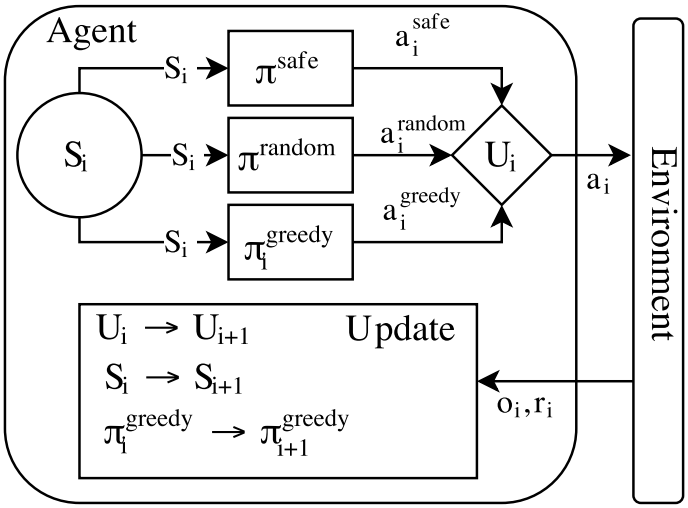


Figure 8.1: ε-safe algorithm.

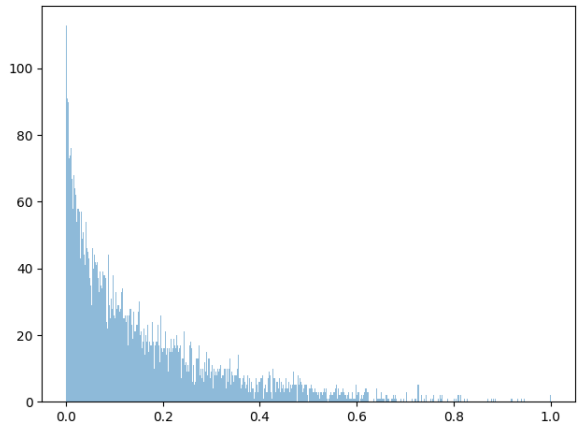


Figure 8.2: Half-Gaussian distribution of p values.

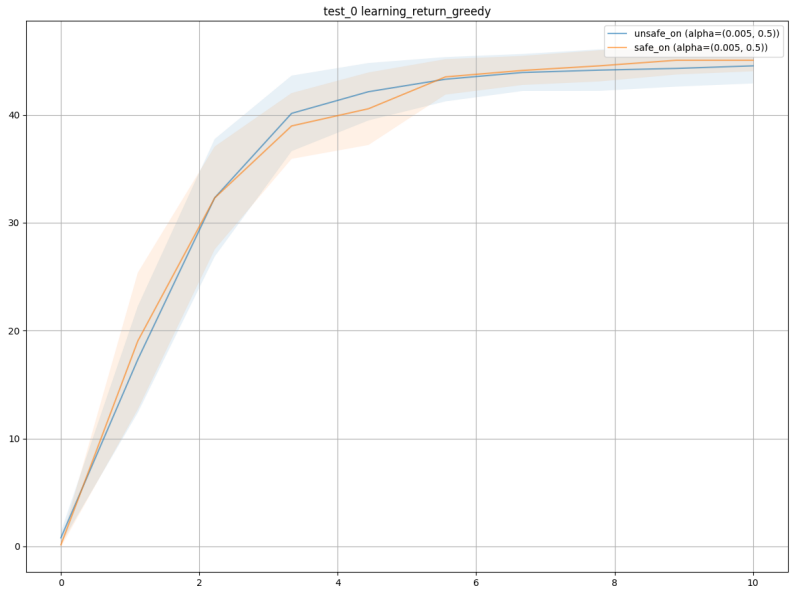
8.3 Experiment

We test our algorithm on a slot-filling application. It is a simpler version of the one use in Chapter 7. The agent asks for slot values, but this time, slot by slot in a fixed order. Several acts are available:

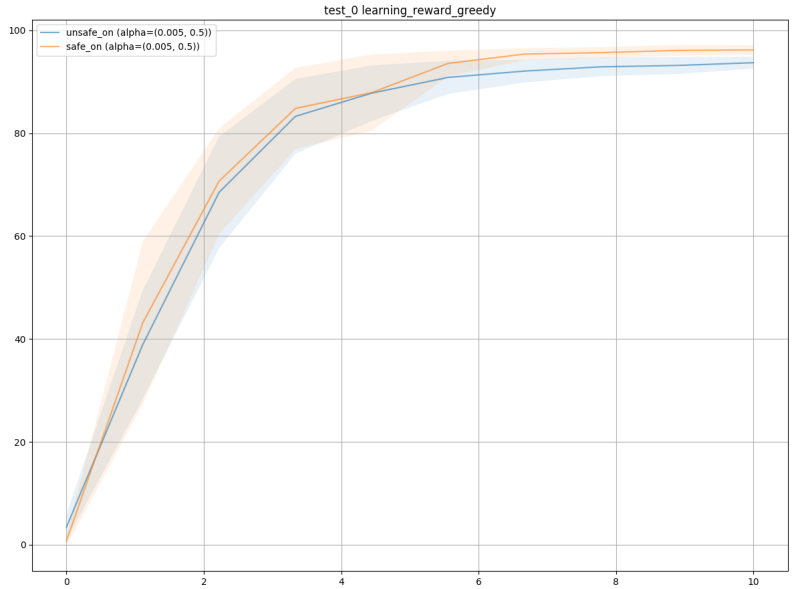
- `ask-next`: ask next slot (with ASR errors).
- `repeat-oral`: repeat current slot (with ASR errors).
- `repeat-numpad`: repeat current slot using numeric pad (without ASR errors).
- `summarize-inform`: summarise slots values and return the form result. If values are correct, the dialogue ends successfully, if not, the slot values are reset and the dialogue continues from the first slot.

`repeat-numpad` is an unsafe action: the user hangups with probability p. For each new user, p is randomly generated. An histogram of p values is displayed Figure 8.2.

We define two handcrafted DSs. The `safe` system uses `repeat-oral` if the recognition score is bellow 0.5, otherwise `ask-next`, or `summarize-inform` after the



(a) Dialogue score



(b) Success frequency

Figure 8.3: Performance of the greedy policies.

last slot; The `unsafe` system uses `repeat-numpad` instead. We compare two ε-safe agents. `safe-on` uses `safe` as transferred policy while `unsafe-on` transfers the `unsafe` policy.

We test `safe-on` and `unsafe-on` with 10 randomly generated users, for 1000 dialogues. We repeat the experiment 10 times. We display the performance of the greedy policies. The dialogue score (reward penalised by dialogue length) is plotted on Figure 8.3a while the dialogue success (reward only) is plotted on Figure 8.3b. We see that despite a slightly difference on the dialogue success, the dialogue score is the same. That means ε-safe does not improve the learning speed of the greedy policy even if it is conservative enough to keep the user in the dialogue by avoiding catastrophic acts.

IV Closing

9 Conclusion 95
9.1 Contributions
9.2 On the long run

10 Appendices 99
.1 Proofs of Main Results
.2 Parameters
.3 Repoducibility

A Continuous transfer in Deep Q-learning 113
A.1 The transfer phase.
A.2 The learning phase
A.3 Conclusion

Bibliography 117

Index 137

9. Conclusion

In this chapter we synthesise the multiple contributions of this manuscript. In Section 9.1, we recall the several limitations of the current approaches for Dialogue Systems (DSs) model and we expose how we tackle those limitations. Then we propose insights to improve or even extend each of those contributions. Finally, in Section 9.2, we foresight the state of art of further works on dialogue systems and user adaptation.

9.1 Contributions

In Chapter 2, we saw that DS applications are booming. In particular, this is the expansion of individual smart phones, TVs and car that makes the research in DS very attractive from the industrial point of view. The current DS technologies in industrial systems mostly rely on handcrafted strategies, but research's state of the art is now based on statistical strategies. For the latter, the Dialogue Manager (DM), defined as the decision making component of a DS (Chapter 3), is usually cast as an Reinforcement Learning (RL) problem (Chapter 4). The strategy is trained from conversational data, improving efficiency, robustness, and facilitating in the meantime the work of the designer. One of the limitations this approach faces is that the dialogue corpus mixes various users records, hence a system that trains on this corpus may adopt a too generic strategy.

To overcome this issue, we propose a solution based on Transfer Learning (TL) (Chapter 5), where we consider a pool of different dialogue corpora corresponding to several users (Chapter 6). The novelty proposed in this work is to introduce a way to cluster the dialogue corpora using the similarities in the behaviour of their respective DS. That way, we can elect representative DSs of the whole database. That allows us to plug the user adaptation framework proposed in Genevay et al. (2016) to operate Online TL. Despite good results with handcrafted users, our experiments suffer from the fact that the toy game we experiment on and the models used are too simple to extract discriminative behaviours among human-model users. One of our further work could consider more challenging

dialogue simulators as for example the complex Negociation Dialogue Game (Laroche 2017). Another path of research could be a direct experiment with real users instead of human-model users. In Appendix A, we will propose to cast this pipeline as a single transfer procedure attached to DQN. Indeed, we believe that extending DQN with TL is of high interest as the dialogue system should continuously improve itself with recurrent users and use previous knowledge to handle a growing base of new users.

Another limitation of current RL approaches for DS is that the very first interactions with a fresh known user are poorly handled. One solution could be using pre-learnt Dialogue Policy (DP) as in Chapter 6. We come up with another idea. In Chapter 7 and Chapter 8, we consider the dialogue process as a safe RL problem where the critical aspect of the dialogue is when a user hangs up. The idea is then to use a generic, but safe policy, as a proxy to the optimal DS, in the first interactions with the new user. This work is divided in two parts.

First, in Chapter 7, we introduce a new batch RL algorithm to learn budgeted policies. We show that those policies managed to contain the hangup frequencies of users bellow a chosen threshold and could be good candidates for proxies. If this approach shown promising results, and theoretical optimality, it lacks of theoretical convergence. We show that it cannot be achieved in the general case. However, we believe that if we restrict the form of the Q-function to a certain form of smoothness actually encountered in the experiments, we can achieve convergence. So the next line of research would involve a deeper theoretical analysis of this convergence. Another issue raised by the algorithm is that it achieves optimally for a restricted class of budgeted policies. Indeed, compared to classical RL, the agent has two degrees of freedom, it can choose the action distribution, and the associated budgets. The problem is, for any next state, the budget given will be the same. We believe that being able to control the budget the next state receives is crucial to achieve optimality over any class of policies. This idea is already explored in (Boutilier et al. 2016) but restricted to known and tractable environment and thus cannot be applied for DSs.

In a second time, in Chapter 8, we introduce an actual method to transfer safe policies. In order to validate the approach in the simplest context we can achieve, instead of learning budgeted policies as proposed in Chapter 7, we design handcrafted policies equipped with different levels of safety in their behaviour. Then, we propose to transfer those policies using the same idea of ε-greedy when the agent alternates between a safe, an exploratory and a greedy policy. Transferring safe policies does not show significant advantage over transferring regular policies. We believe that the hangup-model, based on an uniform distribution, is too simple, so we plan to design the hangup-model with a Poisson distribution. We also want to replace the handcrafted policies by actual RL policies learned on source users using Lagrangian relaxation or even BFTQ. Finally, we may consider a real application on the Dialogue State Tracking Challenge corpus.

9.2 On the long run

The most breathtaking milestones achieved in RL primarily include Deep-Learning solutions. A lot of work has been done for games: OpenAI five (OpenAI 2018) managed to win against the world champion team in Dota 2, a highly competitive Multiplayer Online Battle Arena; DeepMind AlphaGO beat the world champion of GO (Silver et al. 2016), while

AlphaStar (Vinyals et al. 2019) mastered the Real Time Strategy game StarCraft II. While those approaches rely on high-end engineering solutions and an infinity of simulation data, the future of A.I. should focus on end-to-end approaches with no supervision nor plug-n-play simulators (as for real dialogues). The first step in this direction has been done by OpenAI with GPT-2 (Radford et al. 2019), a Transformer language model (Radford et al. 2019) so good that the authors did not release the full model afraid of potential malicious usages. We emphasis on the fact that this model requires no supervision (it is an unsupervised learning algorithm) using attention mechanisms. A very recent attempt tend to incorporate such language models into task-oriented DSs (Budzianowski et al. 2019) and we believe that should be the future of DSs: end-to-end DSs (Serban, Sordoni, et al. 2016; X. Li et al. 2017).

Those DSs will do very well for most of the situation, but will probably fail at user adaptation as those end-to-end approaches do not incorporate a mechanism to differentiate the environments and thus the users. User adaptation will greatly benefit from the recent advances in the robotics field, as one of the main obstacles in RL for robotic is how to transfer the knowledge from the simulated environment to the real environment. We notice an upsurge of work on this subject (Kang et al. 2019; Palossi et al. 2019) and this should continue with the expansion of autonomous driving cars and Unmanned Aerial Vehicules. That being said, the next step in user adaptation should focus on co-adaptation. When dealing with humans, we must expect the user to adapt to the DS. It is particularly true when we want a system dedicated to a specific user as, for example, a vocal assistant. For the moment, the problem has been tackled by a few (Chandramohan et al. 2014; Barlier et al. 2015b).

The last research lead, but not least, should focus on the safety of DSs. In this thesis, one of the first stones was laid, but a lot remains to do. This component is essential to production DSs as nobody wants to face, for instance, a very rude vocal assistant (Perez 2016).

10. Appendices

Outline

This appendix gathers all the supplementary material of Chapter 7 and goes as follows: Appendix .1 details all the proofs of the main results. Appendix .2 lists the parameters used in the experiments. Appendix .3.1 provides instruction to reproduce the experiments. Finally we fill the Machine Learning Reproducibility Checklist and we justify each statement in Appendix .3.2. Finally

.1 Proofs of Main Results

.1.1 Proposition 7.2.1

Proof. This proof is the same as that in classical multi-objective MDPs.

$$
\begin{aligned}
\overline{V}^{\overline{\pi}}(\overline{s}) &\stackrel{\text{def}}{=} \mathbb{E}\left[\overline{G}^{\overline{\pi}} \;\middle|\; \overline{s_0} = \overline{s}\right] \\
&= \sum_{\overline{a} \in \overline{\mathcal{A}}} \mathbb{P}\left(\overline{a}_0 = \overline{a} \mid \overline{s_0} = \overline{s}\right) \mathbb{E}\left[\overline{G}^{\overline{\pi}} \;\middle|\; \overline{s_0} = \overline{s}, \overline{a}_0 = \overline{a}\right] \\
&= \sum_{\overline{a} \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}|\overline{s}) \overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}).
\end{aligned}
$$

$$
\begin{aligned}
\overline{Q}^{\overline{\pi}}(\overline{s},\overline{a}) &\stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\overline{R}(\overline{s}_t,\overline{a}_t)\;\middle|\;\overline{s_0}=\overline{s},\overline{a_0}=\overline{a}\right]\\
&= \overline{R}(\overline{s},\overline{a}) + \sum_{\overline{s}'\in\overline{\mathcal{S}}}\mathbb{P}\left(\overline{s}_1=\overline{s}'\mid\overline{s_0}=\overline{s},\overline{a_0}=\overline{a}\right)\cdot\mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^t\overline{R}(\overline{s}_t,\overline{a}_t)\;\middle|\;\overline{s_1}=\overline{s}'\right]\\
&= \overline{R}(\overline{s},\overline{a}) + \gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}\left(\overline{s}'\mid\overline{s},\overline{a}\right)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\overline{R}(\overline{s}_t,\overline{a}_t)\;\middle|\;\overline{s_0}=\overline{s}'\right]\\
&= \overline{R}(\overline{s},\overline{a}) + \gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}\left(\overline{s}'\mid\overline{s},\overline{a}\right)\overline{V}^{\overline{\pi}}(\overline{s}').
\end{aligned}
$$

Contraction of $\overline{\mathcal{T}}^{\overline{\pi}}$: Let $\overline{\pi}\in\overline{\Pi},\overline{Q}_1,\overline{Q}_2\in(\mathbb{R}^2)^{\overline{\mathcal{S}}\overline{\mathcal{A}}}$.

$$
\begin{aligned}
\forall\overline{s}\in\overline{\mathcal{S}},\overline{a}\in\overline{\mathcal{A}},\quad \left|\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_1(\overline{s},\overline{a})-\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_2(\overline{s},\overline{a})\right| &= \left|\gamma\mathop{\mathbb{E}}_{\substack{\overline{s}'\sim\overline{P}(\overline{s}'|\overline{s},\overline{a})\\ \overline{a}'\sim\overline{\pi}(\overline{a}'|\overline{s}')}}\overline{Q}_1(\overline{s}',\overline{a}')-\overline{Q}_2(\overline{s}',\overline{a}')\right|\\
&\leq\gamma\left\|\overline{Q}_1-\overline{Q}_2\right\|_\infty.
\end{aligned}
$$

Hence, $\left\|\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_1-\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_2\right\|_\infty\leq\gamma\left\|\overline{Q}_1-\overline{Q}_2\right\|_\infty$

According to the Banach fixed point theorem, $\overline{\mathcal{T}}^{\overline{\pi}}$ admits a unique fixed point. It can be easily verified that $\overline{Q}^{\overline{\pi}}$ is indeed this fixed point by combining the two Bellman Expectation equations (7.6). ■

.1.2 Theorem 7.2.2

Proof. Let $\overline{s},\overline{a}\in\overline{\mathcal{A}}\times\overline{\mathcal{S}}$. For this proof, we consider potentially non-stationary policies $\overline{\pi}=(\rho,\overline{\pi}')$, with $\rho\in\mathcal{M}(\overline{\mathcal{A}})$, $\overline{\pi}'\in\mathcal{M}(\overline{\mathcal{A}})^{\mathbb{N}}$. The results will apply to the particular case of stationary optimal policies, when they exist.

$$
\begin{aligned}
{Q_r}^*(\overline{s},\overline{a}) &= \max_{\rho,\overline{\pi}'}{Q_r}^{\rho,\overline{\pi}'}(\overline{s}',\overline{a}') && (1)\\
&= \max_{\rho,\overline{\pi}'}R(s,a)+\gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}(\overline{s}'|\overline{s},\overline{a})V_r^{\rho,\overline{\pi}'}(\overline{s}') && (2)\\
&= R(s,a)+\gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}(\overline{s}'|\overline{s},\overline{a})\max_{\rho,\overline{\pi}'}\sum_{\overline{a}'\in\overline{\mathcal{A}}}\rho(\overline{a}'|\overline{s}'){Q_r}^{\overline{\pi}'}(\overline{s}',\overline{a}') && (3)\\
&= R(s,a)+\gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}(\overline{s}'|\overline{s},\overline{a})\max_{\rho}\sum_{\overline{a}'\in\overline{\mathcal{A}}}\rho(\overline{a}'|\overline{s}')\max_{\overline{\pi}'\in\overline{\Pi}_a(\overline{s}')}{Q_r}^{\overline{\pi}'}(\overline{s}',\overline{a}') && (4)\\
&= R(s,a)+\gamma\sum_{\overline{s}'\in\overline{\mathcal{S}}}\overline{P}(\overline{s}'|\overline{s},\overline{a})\max_{\rho}\mathop{\mathbb{E}}_{\overline{a}'\sim\rho}{Q_r}^*(\overline{s}',\overline{a}') && (5)
\end{aligned}
$$

where $\overline{\pi}=(\rho,\overline{\pi}')\in\overline{\Pi}_a(\overline{s})$ and $\overline{\pi}'\in\overline{\Pi}_a(\overline{s}')$.

This follows from:

(1). Definition of $\overline{Q}^*$.

(2). Bellman Expectation expansion from Proposition 7.2.1.

(3). Marginalisation on $\overline{a}'$.

(4).
- Trivially $\max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} \sum_{\overline{a}' \in \overline{\mathcal{A}}} \cdot \leq \sum_{\overline{a}' \in \overline{\mathcal{A}}} \max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s})} \cdot$.
- Let $\overline{\overline{\pi}} \in \arg\max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} {Q_r}^{\overline{\pi}'}(\overline{s}', \overline{a}')$, then:

$$\sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') \max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} {Q_r}^{\overline{\pi}'}(\overline{s}', \overline{a}') = \sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') {Q_r}^{\overline{\overline{\pi}}}(\overline{s}', \overline{a}')$$
$$\leq \max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} \sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') {Q_r}^{\overline{\pi}'}(\overline{s}', \overline{a}').$$

(5). Definition of $\overline{Q}^*$.
Moreover, the condition $\overline{\pi} = (\rho, \overline{\pi}') \in \overline{\Pi}_a(\overline{s})$ gives

$$\mathop{\mathbb{E}}_{\overline{a}' \sim \rho} {Q_c}^*(\overline{s}, \overline{a}) = \mathop{\mathbb{E}}_{\overline{a}' \sim \rho} {Q_c}^{\overline{\pi}'}(\overline{s}, \overline{a}) = {V_c}^{\overline{\pi}}(\overline{s}) \leq \beta.$$

Consequently, $\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)$ belongs to the $\arg\max$ of (5), and in particular:

$${Q_r}^*(\overline{s}, \overline{a}) = r(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} P(\overline{s}'|\overline{s}, \overline{a}) \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{s}', \overline{Q}^*)} {Q_r}^*(\overline{s}', \overline{a}').$$

The same reasoning can be made for ${Q_c}^*$ by replacing $\max$ operators by $\min$, and $\overline{\Pi}_a$ by $\overline{\Pi}_r$.

■

.1.3 Proposition 7.2.3

Proof. Notice from the definitions of $\overline{\mathcal{T}}^*$ and $\overline{\mathcal{T}}^{\overline{\pi}}$ in (7.12) and (7.7) that $\overline{\mathcal{T}}^*$ and $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)}$ coincide on $\overline{Q}^*$. Moreover, since $\overline{Q}^* = \overline{\mathcal{T}}^* \overline{Q}^*$ by Theorem 7.2.2, we have: $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)} \overline{Q}^* = \overline{\mathcal{T}}^* \overline{Q}^* = \overline{Q}^*$. Hence, $\overline{Q}^*$ is a fixed point of $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)}$, and by Proposition 7.2.1 it must be equal to $\overline{Q}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^*)}$

To show the same result for $\overline{V}^*$, notice that

$$\overline{V}^{\overline{\pi}_{\text{greedy}}(\overline{Q}^*)}(\overline{s}) = \mathop{\mathbb{E}}_{\overline{a} \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^*)} \overline{Q}^{\overline{\pi}_{\text{greedy}}(\overline{Q}^*)}(\overline{s}, \overline{a}) = \mathop{\mathbb{E}}_{\overline{a} \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^*)} \overline{Q}^*(\overline{s}, \overline{a}).$$

By applying the definitions of $\overline{Q}^*$ and $\overline{\pi}_{\text{greedy}}$, we recover the definition of $\overline{V}^*$. ■

.1.4 Theorem 7.2.4

Proof. In the trivial case $|\mathcal{A}| = 1$, there exits only one policy $\overline{\pi}$ and $\overline{\mathcal{T}}^* = \overline{\mathcal{T}}^{\overline{\pi}}$, which is a contraction by Proposition 7.2.1.

In the general case $|\mathcal{A}| \geq 2$, we can build the following counter-example:

Let $(\mathcal{S}, \mathcal{A}, P, R, C)$ be a BMDP. For any $0 < \kappa < 1$, we define $\overline{Q}^1_\kappa$ and $\overline{Q}^2_\kappa$ as:

$$\overline{Q}^1_\kappa(\overline{s}, \overline{a}) = \begin{cases} (0, 1), & \text{if } a = a_0 \\ (\frac{1}{\kappa}, 1 + \kappa), & \text{if } a \neq a_0 \end{cases}$$
$$\overline{Q}^2_\kappa(\overline{s}, \overline{a}) = \begin{cases} (1, 0), & \text{if } a = a_0 \\ (1 + \frac{1}{\kappa}, \kappa), & \text{if } a \neq a_0 \end{cases}$$

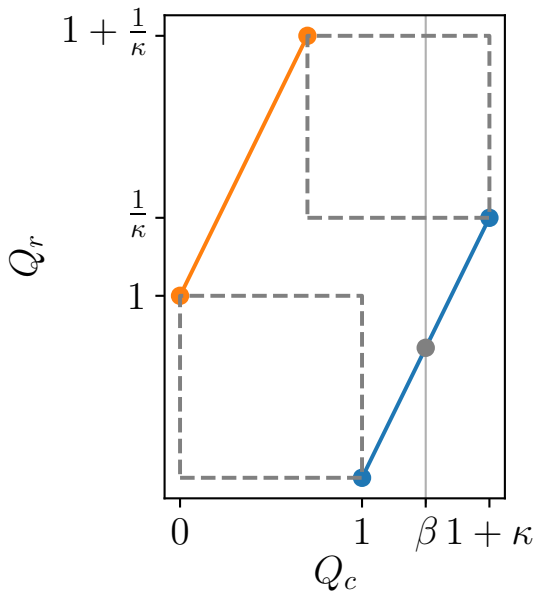


Figure 1: Representation of $\overline{Q}^1_\kappa$ (blue) and $\overline{Q}^2_\kappa$ (orange)

Then, $\|\overline{Q}_1 - \overline{Q}_2\|_\infty = 1$. $\overline{Q}^1_\kappa$ and $\overline{Q}^2_\kappa$ are represented in Figure 1.

But for $\overline{a} = (a, \beta_a)$ with $\beta_a = 1 + \kappa/2$, we have:

$$\begin{aligned}
&\|\overline{\mathcal{T}}^*\overline{Q}^1_\kappa(\overline{s},\overline{a}) - \overline{\mathcal{T}}^*\overline{Q}^2_\kappa(\overline{s},\overline{a})\|_\infty \\
&\qquad = \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}'\sim\overline{P}(\overline{s}'|\overline{s},\overline{a})} \mathop{\mathbb{E}}_{\overline{a}'\sim\overline{\pi}_{\text{greedy}}(\overline{Q}^1_\kappa)} \overline{Q}^1_\kappa(\overline{s}',\overline{a}') - \mathop{\mathbb{E}}_{\overline{a}'\sim\overline{\pi}_{\text{greedy}}(\overline{Q}^2_\kappa)} \overline{Q}^2_\kappa(\overline{s}',\overline{a}') \right\|_\infty \\
&\qquad = \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}'\sim\overline{P}(\overline{s}'|\overline{s},\overline{a})} (\frac{1}{2\kappa}, 1+\frac{\kappa}{2}) - (1+\frac{1}{\kappa}, \kappa) \right\|_\infty \\
&\qquad = \gamma(1+\frac{1}{2\kappa}).
\end{aligned}$$

Hence, $\|\overline{\mathcal{T}}^*\overline{Q}^1_\kappa - \overline{\mathcal{T}}^*\overline{Q}^2_\kappa\|_\infty \geq \gamma(1+\frac{1}{2\kappa})$.

As a conclusion, there does not exist $L > 0$ such that:

$$\forall \overline{Q}_1, \overline{Q}_2 \in (\mathbb{R}^2)^{\overline{\mathcal{SA}}}, \|\overline{\mathcal{T}}^*\overline{Q}^1 - \overline{\mathcal{T}}^*\overline{Q}^2\|_\infty \leq L\|\overline{Q}^1 - \overline{Q}^2\|_\infty$$

In other words, $\overline{\mathcal{T}}^*$ is not a contraction for $\|\cdot\|_\infty$. ■

.1.5 Theorem 7.2.5

R This proof makes use of insights detailed in the proof of Proposition 7.4.1 (Appendix .1.6), which we recommend the reader to consult first.

Proof. We now study the contractivity of $\overline{\mathcal{T}}^*$ when restricted to the functions of $\mathcal{L}_\gamma$ defined as follows:

$$\mathcal{L}_\gamma = \left\{ \begin{array}{l} \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{SA}}} \text{ s.t. } \exists L < \frac{1}{\gamma} - 1 : \forall \overline{s} \in \overline{\mathcal{S}}, \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \\ |Q_r(\overline{s},\overline{a}_1) - Q_r(\overline{s},\overline{a}_2)| \leq L|Q_c(\overline{s},\overline{a}_1) - Q_c(\overline{s},\overline{a}_2)| \end{array} \right\}. \tag{6}$$

That is, for all state $\overline{s}$, the set $\overline{Q}(\overline{s}, \overline{\mathcal{A}})$ plot in the (Q_c, Q_r) plane must be the *graph* of a L-Lipschitz function, with $L < 1/\gamma - 1$.

We impose such structure for the following reason: the counter-example presented above prevented contraction because it was a pathological case in which the slope of $\overline{Q}$ can be arbitrary large. As a consequence, when solving ${Q_r}^*$ such that ${Q_c}^* = \beta$, a vertical slice of a $\|\cdot\|_\infty$ ball around $\overline{Q}_1$ (which must contain $\overline{Q}_2$) can be arbitrary large as well.

We denote $\text{Ball}(\overline{Q}, R)$ the ball of centre $\overline{Q}$ and radius R for the $\|\cdot\|_\infty$-norm:

$$\text{Ball}(\overline{Q}, R) = \{\overline{Q}' \in (R^2)^{\overline{S}\overline{\mathcal{A}}} : \|\overline{Q} - \overline{Q}'\|_\infty \leq R\}.$$

We give the three main steps required to show that $\overline{\mathcal{T}}^*$ restricted to $\mathcal{L}_\gamma$ is a contraction. Given $\overline{Q}^1, \overline{Q}^2 \in \mathcal{L}_\gamma$, show that:

1. $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R) \implies \mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R), \forall \overline{s} \in \overline{\mathcal{S}}$, where $\mathcal{F}$ is the top frontier of the convex hull of undominated points, as defined in Appendix .1.6.
2. $\overline{Q} \in \mathcal{L}_\gamma \implies \mathcal{F}$ is the graph of a L-Lipschitz function, $\forall \overline{s} \in \overline{\mathcal{S}}$.
3. taking the slice $Q_c = \beta$ of a ball $\text{Ball}(\mathcal{F}, R)$ with $\mathcal{F}$ L-Lipschitz results in an interval on Q_r of range at most $(L+1)R$

These three steps will allow us to control ${Q_r}^{2*} - {Q_r}^{1*}$ as a function of $R = \|\overline{Q}^2 - \overline{Q}^1\|_\infty$.

Step 1

We want to show that if $\overline{Q}^1$ and $\overline{Q}^2$ are close, then $\mathcal{F}^1$ are $\mathcal{F}^2$ are close as well in the following sense:

$$\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R) \iff d(\mathcal{F}^1, \mathcal{F}^2) \leq R \iff \max_{q^2 \in \mathcal{F}^2} \min_{q^1 \in \mathcal{F}^1} \|q^2 - q^1\|_\infty \leq R. \quad (7)$$

Assume $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R)$, we show by contradiction that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$. Indeed, assume there exists $q^1 \in \mathcal{F}^1$ such that $\mathcal{F}^2 \cap \text{Ball}(q^1, R) = \emptyset$. Denote q^2 the unique point of $\mathcal{F}^2$ such that $q^2_c = q^1_c$. By construction of q^1, we know that $\|q^1 - q^2\|_\infty > R$. There are two possible cases:

- $q^2_r > q^1_r$: this also directly implies that $q^2_r > q^1_r + R$. But $q^2 \in \mathcal{F}^2$, so there exist $q^2_1, q^2_2 \in Q^2, \lambda \in \mathbb{R}$ such that $q^2 = (1-\lambda)q^2_1 + \lambda q^2_2$. But since $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R)$, there also exist $q^1_1, q^1_2 \in \overline{Q}^1$ such that $\|q^1_1 - q^2_1\|_\infty \leq R$ and $\|q^1_2 - q^2_2\|_\infty \leq R$, and in particular $q^1_{1r} \geq q^2_{1r} - R$ and $q^1_{2r} \geq q^2_{2r} - R$. But then, the point $q^{1'} = (1-\mu)q^1_1 + \mu q^1_2$ with $\mu = (q^2_c - q^1_{1c})/(q^2_{2c} - q^1_{1c})$ verifies $q^{1'}_c = q^1_c$ and $q^{1'}_r \geq q^2_r - R > q^1_r$ which contradicts the definition of $q_1 \in \mathcal{F}^1$ as defined in (12).
- $q^2_r < q^1_r$: then the same reasoning can be applied by simply swapping the indexes 1 and 2.

We have shown that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$. This is illustrated in Figure 2: given a function $\overline{Q}^1$, we show the locus $\text{Ball}(\overline{Q}_1, R)$ of $\overline{Q}^2$. We then draw $\mathcal{F}^1$ the top frontier of the convex hull of $\overline{Q}^1$ and alongside the locus of all possible $\mathcal{F}^2$, which belong to a ball $\text{Ball}(\mathcal{F}^1, R)$.

Step 2

We want to show that if $\overline{Q} \in \mathcal{L}_\gamma$, $\mathcal{F}$ is the graph of an L-Lipschitz function:

$$\forall q^1, q^2 \in \mathcal{F}, |q^2_r - q^1_r| \leq |q^2_c - q^1_c|. \quad (8)$$

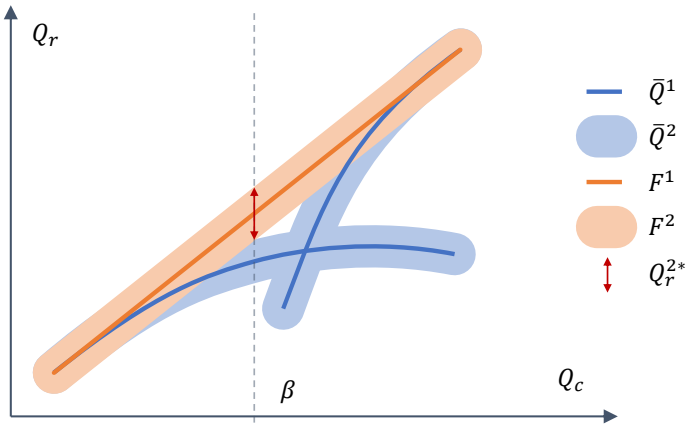


Figure 2: We represent the range of possible solutions ${Q_r}^{2,*}$ for any $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1)$, given $\overline{Q}_1 \in \mathcal{L}_\lambda$

Let $\overline{Q} \in \mathcal{L}_\gamma$ and $\overline{s} \in \overline{\mathcal{S}}$, $\mathcal{F}$ the corresponding top frontier of convex hull. For all $q^1, q^2 \in \mathcal{F}, \exists \lambda, \mu \in [0,1], q^{11}, q^{12}, q^{21}, q^{22} \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})$ such that $q^1 = (1-\lambda)q^{11} + \lambda q^{12}$ and $q^2 = (1-\mu)q^{21} + \mu q^{22}$. Without loss of generality, we can assume $q_c^{11} \leq q_c^{12}$ and $q_c^{21} \leq q_c^{22}$. We also consider the worst case in terms of maximum q_r deviation: $q_c^{12} \leq q_c^{21}$. Then the maximum increment $q_r^2 - q_r^1$ is:

$$
\begin{aligned}
\|q_r^2 - q_r^1\| &\leq \|q_r^{12} - q_r^1\| + \|q_r^{21} - q_r^{12}\| + \|q_r^2 - q_r^{21}\| \\
&= (1-\lambda)\|q_r^{12} - q_r^{11}\| + \|q_r^{21} - q_r^{12}\| + \mu\|q_r^{22} - q_r^{21}\| \\
&\leq (1-\lambda)L\|q_c^{12} - q_c^{11}\| + L\|q_c^{21} - q_c^{12}\| + \mu L\|q_c^{22} - q_c^{21}\| \\
&= L\|q_c^{12} - q_c^1\| + L\|q_c^{21} - q_c^{12}\| + L\|q_c^2 - q_c^{21}\| \\
&= L\|q_c^2 - q_c^1\|.
\end{aligned}
$$

This can also be seen in Figure 2: the maximum slope of the $\mathcal{F}^1$ is lower than the maximum slope between two points of $\overline{Q}^1$.

Step 3

Let $\mathcal{F}_1$ be a L-Lipschitz set as defined in (8), and consider a ball $\text{Ball}(\mathcal{F}_1, R)$ around it as defined in (7).

We want to bound the optimal reward value ${Q_r}^{2*}$ under constraint ${Q_c}^{2*} = \beta$ (regular case in Appendix .1.6 where the constraint is saturated), for any $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}_1, R)$. This quantity is represented as a red double-ended arrow in Figure 2.

Because we are only interested in what happens locally at $Q_c = \beta$, we can zoom in on Figure 2 and only consider a thin κ-section around β. In the limit $\kappa \to 0$, this section becomes the tangent to $\mathcal{F}^1$ at ${Q_c}^1 = \beta$. It is represented in Figure 3, from which we derive a geometrical proof:

$$
\begin{aligned}
\Delta {Q_r}^{2*} &= b + c \\
&\leq La + c && (\mathcal{F}^1\ L\text{-Lipschitz}) \\
&= 2LR + 2R = 2R(L+1).
\end{aligned}
$$

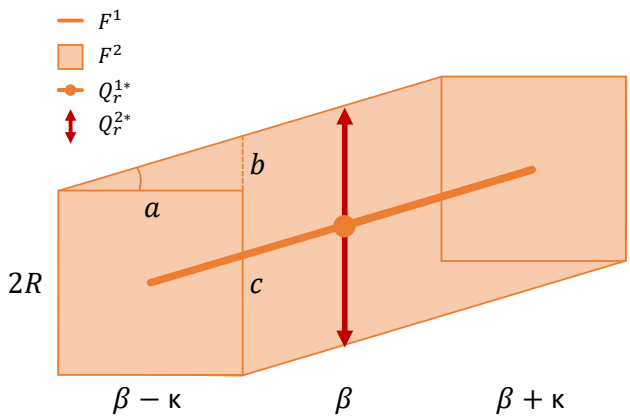


Figure 3: We represent a section $[\beta - \kappa, \beta + \kappa]$ of $\mathcal{F}^1$ and $\text{Ball}(\mathcal{F}^1, R)$. We want to bound the range of ${Q_r}^{2*}$.

Hence,

$$|{Q_r}^{2*} - {Q_r}^{1*}| \leq \frac{\Delta {Q_r}^{2*}}{2} = R(L+1)$$

and ${Q_c}^{1*} = {Q_c}^{2*} = \beta$. Consequently, $\|\overline{Q}^{2*} - \overline{Q}^{1*}\|_\infty \leq (L+1)R$.

Finally, consider the edge case in Appendix .1.6: the constraint is not active, and the optimal value is simply $\arg\max_{q \in \mathcal{F}} q^r$. In particular, since we showed that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$, and since $\overline{Q}^{2*} \in \mathcal{F}^2$, there exist $q^1 \in \mathcal{F}^1 : \|\overline{Q}^{2*} - q^1\|_\infty \leq R$ and in particular $\overline{Q}_r^{1*} \geq q_r^1 \geq \overline{Q}_r^{2*} - R$. Reciprocally, by the same reasoning, ${Q_r}^{2*} \geq {Q_r}^{1*} - R$. Hence, we have that $|{Q_r}^{2*} - {Q_r}^{1*}| \leq R \leq R(L+1)$.

Wrapping it up

We have shown that for any $\overline{Q}^1, \overline{Q}^2 \in \mathcal{L}_\gamma$, and all $\overline{s} \in \overline{\mathcal{S}}$, $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, \|\overline{Q}^2 - \overline{Q}^1\|_\infty)$ and $\mathcal{F}^1$ is the graph of a L-Lipschitz function with $L < 1/\gamma - 1$. Moreover, the solutions of $\overline{\pi}_{\text{greedy}}(\overline{Q}^1)$ and $\overline{\pi}_{\text{greedy}}(\overline{Q}^2)$ at $\overline{s}$ are such that $\|\overline{Q}^{2*} - \overline{Q}^{1*}\|_\infty \leq (L+1)\|\overline{Q}^2 - \overline{Q}^1\|_\infty$.

Hence, for all $\overline{a}$,

$$\begin{aligned}
&\|\overline{\mathcal{T}}^*\overline{Q}^1(\overline{s}, \overline{a}) - \overline{\mathcal{T}}^*\overline{Q}^2(\overline{s}, \overline{a})\|_\infty \\
&\qquad = \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}' \sim \overline{P}(\overline{s}'|\overline{s}, \overline{a})} \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^1)} \overline{Q}^1(\overline{s}', \overline{a}') - \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^2)} \overline{Q}^2(\overline{s}', \overline{a}') \right\|_\infty \\
&\qquad = \gamma \left\| \overline{Q}^{2*} - \overline{Q}^{1*} \right\|_\infty \\
&\qquad \leq \gamma(L+1)\|\overline{Q}^2 - \overline{Q}^1\|_\infty.
\end{aligned}$$

Taking the sup on $\overline{\mathcal{SA}}$,

$$\|\overline{\mathcal{T}}^*\overline{Q}^1 - \overline{\mathcal{T}}^*\overline{Q}^2\|_\infty \leq \gamma(L+1)\|\overline{Q}^1 - \overline{Q}^2\|_\infty$$

with $\gamma(L+1) < 1$. As a conclusion, $\overline{\mathcal{T}}^*$ is a $\gamma(L+1)$-contraction on $\mathcal{L}_\gamma$. ■

.1.6 Proposition 7.4.1

Definition .1.1 Let A be a set, and f a function defined on A. We define:

- Convex hull of A: $\mathcal{C}(A) = \{\sum_{i=1}^{p} \lambda_i a_i : a_i \in A, \lambda_i \in \mathbb{R}^+, \sum_{i=1}^{p} \lambda_i = 1, p \in \mathbb{N}\}$
- Convex edges of A: $\mathcal{C}^2(A) = \{\lambda a_1 + (1-\lambda) a_2 : a_1, a_2 \in A, \lambda \in [0,1]\}$
- Dirac distributions of A: $\delta(A) = \{\delta(a - a_0) : a_0 \in A\}$
- Image of A by f: $f(A) = \{f(a) : a \in A\}$

Proof. Let $\overline{s} = (s, \beta) \in \overline{\mathcal{S}}$ and $\overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}}\overline{\mathcal{A}}}$. We recall the definition of $\overline{\pi}_{\text{greedy}}$:

$$\overline{\pi}_{\text{greedy}}(\overline{a}|\overline{s}; \overline{Q}) \in \arg\min_{\rho \in \overline{\Pi}_r^{\overline{Q}}} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s}, \overline{a}) \tag{7.13a}$$

$$\text{where} \quad \overline{\Pi}_r^{\overline{Q}} = \arg\max_{\rho \in \mathcal{M}(\overline{\mathcal{A}})} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_r(\overline{s}, \overline{a}) \tag{7.13b}$$

$$\text{s.t.} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s}, \overline{a}) \leq \beta \tag{7.13c}$$

Note that any policy in the $\arg\min$ in (7.13a) is suitable to compute $\overline{\mathcal{T}}^*$. We first reduce the set of candidate optimal policies. Consider the problem described in (7.13b),(7.13c): it can be seen as a single-step CMDP problem with reward $R = Q_r$ and cost $C = Q_c$. By (Theorem 4.4 Beutler et al. 1985), we know that the solutions are mixtures of two deterministic policies. Hence, we can replace $\mathcal{M}(\mathcal{A})$ by $\mathcal{C}^2(\delta(\overline{\mathcal{A}}))$ in (7.13b).

Moreover, remark that:

$$\begin{aligned}
&\{\mathop{\mathbb{E}}_{\overline{a}\sim\rho} \overline{Q}(\overline{s}, \overline{a}) : \rho \in \mathcal{C}^2(\delta(\overline{\mathcal{A}}))\} \\
&\qquad = \{\mathop{\mathbb{E}}_{\overline{a}\sim\rho} \overline{Q}(\overline{s}, \overline{a}) : \rho = (1-\lambda)\delta(\overline{a} - \overline{a}_1) + \lambda\delta(\overline{a} - \overline{a}_2), \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \lambda \in [0,1]\} \\
&\qquad = \{(1-\lambda)\overline{Q}(\overline{s}, \overline{a}_1) + \lambda\overline{Q}(\overline{s}, \overline{a}_2), \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \lambda \in [0,1]\} \\
&\qquad = \mathcal{C}^2(\overline{Q}(\overline{s}, \overline{\mathcal{A}}))\}.
\end{aligned}$$

Hence, the problem (7.13b), (7.13c) has become:

$$\overline{\widetilde{\Pi}}^{Q_r} = \arg\max_{(q_r, q_c) \in \mathcal{C}^2(\overline{Q}(\overline{s}, \overline{\mathcal{A}}))} q_r \quad \text{s.t.} \quad q_c \leq \beta$$

and the solution of $\overline{\pi}_{\text{greedy}}$ is $q^* = \arg\min_{q \in \overline{\widetilde{\Pi}}^{Q_r}} q_c$.

The original problem in the space of actions $\overline{\mathcal{A}}$ is now expressed in the space of values $\overline{Q}(\overline{s}, \overline{\mathcal{A}})$ (which is why we use $=$ instead of $\in$ before $\arg\min$ here).

We further restrict the search space of q^* following two observations:

1. q^* belongs to the *undominated* points $\mathcal{C}^2(\overline{Q}^-)$:

$$\overline{Q}^+ = \{(q_c, q_r) : q_c > q_c^{\pm} = \min_{q^+} q_c^+ \text{ s.t. } q^+ \in \arg\max_{q \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})} q_r\} \tag{10}$$

$$\overline{Q}^- = \overline{Q}(\overline{s}, \overline{\mathcal{A}}) \setminus \overline{Q}^+. \tag{11}$$

Denote $q^* = (1-\lambda)q^1 + \lambda q^2$, with $q^1, q^2 \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})$. There are three possible cases:

(a) $q^1, q^2 \notin \overline{Q}^-$. Then $q_c^* = (1-\lambda)q_c^1 + \lambda q_c^2 > q_c^\pm$. But then $q_c^\pm < q_c^* \leq \beta$ so $q^\pm \in \overset{\sim}{\Pi}^{Q_r}$ with a strictly lower q_c than q^*, which contradicts the $\arg\min$.

(b) $q^1 \in \overline{Q}^-, q^2 \notin \overline{Q}^-$. But then consider the mixture $q^\top = (1-\lambda)q^1 + \lambda q^\pm$. Since $q_r^\pm \geq q_r^2$ and $q_r^\pm < q_r^2$, we also have $q_r^\top \geq q_r^*$ and $q_c^\top < q_c^*$, which also contradicts the $\arg\min$.

(c) $q^1, q^2 \in \overline{Q}^-$ is the only remaining possibility.

2. q^* belongs to the *top frontier* $\mathcal{F}$:

$$\mathcal{F}_{\overline{Q}} = \{q \in \mathcal{C}^2(\overline{Q}^-) : \nexists q' \in \mathcal{C}^2(\overline{Q}^-) : q_c = q'_c \text{ and } q_r < q'_r\}. \tag{12}$$

Trivially, otherwise q' would be a better candidate than q^*.

Let us characterise this frontier $\mathcal{F}$. It is both:

1. the *graph of a non-decreasing function*: $\forall q^1, q^2 \in \mathcal{F}$ such that $q_c^1 \leq q_c^2$ then $q_r^1 \leq q_r^2$. By contradiction, if we had $q_r^1 > q_r^2$, we could define $q^\top = (1-\lambda)q^1 + \lambda q^\pm$ where $q^\pm$ is the dominant point as defined in (10). By choosing $\lambda = (q_c^2 - q_c^1)/(q_c^\pm - q_c^1)$ such that $q_c^\top = q_c^2$, then since $q_r^\pm \geq q_r^1 > q_r 2$ we also have $q_r^\top > q_r^2$ which contradicts $q^2 \in \mathcal{F}$.
2. the *graph of a concave function*: $\forall q^1, q^2, q^3 \in \mathcal{F}$ such that $q_c^1 \leq q_c^2 \leq q_c^3$ with λ such that $q_c^2 = (1-\lambda)q_c^1 + \lambda q_c^3$, then $q_r^2 \geq (1-\lambda)q_r^1 + \lambda q_r^3$.
   Trivially, otherwise the point $q^\top = (1-\lambda)q^1 + \lambda q^3$ would verify $q_c^\top = q_c^2$ and $q_r^\top > q_r^2$, which would contradict $q^2 \in \mathcal{F}$.

We denote $\mathcal{F}_{\overline{Q}} = \mathcal{F} \cap \overline{Q}$. Clearly, $q^* \in \mathcal{C}^2(\mathcal{F}_{\overline{Q}})$: let $q^1, q^2 \in \overline{Q}^-$ such that $q^* = (1-\lambda)q^1 + \lambda q^2$. First, $q^1, q^2 \in \overline{Q}^- \subset \mathcal{C}^2(\overline{Q}^-)$. Then, by contradiction, if there existed $q^{1'}$ or $q^{2'}$ with equal q_c and strictly higher q_r, again we could build an admissible mixture $q^\top = (1-\lambda)q^{1'} + \lambda q^{2'}$ strictly better than q^*.

q^* can be written as $q^* = (1-\lambda)q^1 + \lambda q^2$ with $q^1, q^2 \in \mathcal{F}_{\overline{Q}}$ and, without loss of generality, $q_c^1 \leq q_c^2$.

Regular case

There exists $q^0 \in \mathcal{F}_{\overline{Q}}$ such that $q_c^0 \geq \beta$. Then q^1 and q^2 must flank the budget: $q_c^1 \leq \beta \leq q_c^2$. Indeed, by contradiction, if $q_c^2 \geq q_c^1 > \beta$ then $q_c^* > \beta$ which contradicts $\overline{\Pi}_r^{\overline{Q}}$. Conversely, if $q_c^1 \leq q_c^2 < \beta$ then $q^* < \beta \leq q_c^0$, which would make q^* a worse candidate than $q^\top = (1-\lambda)q^* + \lambda q^0$ when λ is chosen such that $q_c^\top = \beta$, and contradict $\overline{\Pi}_r^{\overline{Q}}$ again.

Because $\mathcal{F}$ is the graph of a non-decreasing function, λ should be as high as possible, as long as the budget $q^* \leq \beta$ is respected. We reach the highest q_r^* when $q_c^* = \beta$, that is: $\lambda = (\beta - q_c^1)/(q_c^2 - q_c^1)$.

It remains to show that q^1 and q^2 are two successive points in $\mathcal{F}_{\overline{Q}}$: $\nexists q \in \mathcal{F}_{\overline{Q}} \setminus \{q^1, q^2\} : q_c^1 \leq q_c \leq q_c^2$. Otherwise, as $\mathcal{F}$ is the graph of a concave function, we would have $q_r \geq (1-\mu)q_r^1 + \mu q_r^2$. q_r cannot be strictly greater than $(1-\mu)q_r^1 + \mu q_r^2$ which would contradict q^*, but it can still be equal, which means the tree points q, q^1, q^2 are aligned. In fact, every points aligned with q^1 and q^2 can also be used to construct mixtures resulting in q^*, but among these solutions we can still choose q^1 and q^2 as the two points in $\mathcal{F}_{\overline{Q}}$ closest to q^*.

Edge case

$\forall q \in \mathcal{F}_{\overline{Q}}, q_c < \beta$. Then $q^* = \arg\max_{q \in \mathcal{F}} q_r = q^\pm = \arg\max_{q \in \overline{Q}^-} q_r$. ■

Parameters	BFTQ(risk-sensitive)	BFTQ(risk-neutral)
architecture	256x128x64	256x128x64
regularisation	0.001	0.001
activation	relu	relu
size beta encoder	3	3
initialisation	xavier	xavier
loss function	L2	L2
optimizer	adam	adam
learning rate	0.001	0.001
epoch (NN)	1000	5000
normalize reward	true	true
epoch (FTQ)	12	12
$\widetilde{\mathscr{B}}$	0:0.01:1	-
γ	1	1
$N = \lvert\mathcal{D}\rvert$	5000	5000
$N_{\text{minibatch}}$	10	10
N_{seeds}	4	4
N_{test}	1000	1000
decay epsilon scheduling	0.001	0.001

Table 1: Algorithms parameters for `Corridors`

.2 Parameters

All algorithm parameters are displayed in Table 1,Table 2 and Table 3.

State-Space

The states s (from $\overline{s} = (s, \beta)$) of the agent are described in the following:

- `Corridors`: $s = (x, y)$ where x and y are the 2D coordinates of the agent.
- `slot-filling`: $s = (\nu, \text{min}, a_u, a_s, t)$ where ν is a vector of the SRS for each slot, min is a one hot vector describing the minimum of the ν vector, a_u is a one hot vector of the last user dialogue act and a_s is the one hot vector of the last system dialogue act. Finally $t \in [0, 1]$ is the fraction of the current turn with the maximum number of turns authorised.
- `Highway-Env`: the positions (x, y) and velocities $(\dot{x}, \dot{y})$ of every vehicle on the road.

A note on the parameters search

We performed a shallow grid-search for the classic NN parameters. Most of the parameters do not have a strong influence on the results, however in the `slot-filling` environment, the choice of the regulation weight is decisive.

Parameters	BFTQ	FTQ
architecture	256x128x64	128x64x32
regularisation	0.0005	0.0005
activation	relu	relu
size beta encoder	50	-
initialisation	xavier	xavier
loss function	L2	L2
optimizer	adam	adam
learning rate	0.001	0.001
epoch (NN)	5000	5000
normalize reward	true	true
epoch (FTQ)	11	11
$\widetilde{\mathscr{B}}$	0:0.01:1	-
γ	1	1
$N = \lvert\mathcal{D}\rvert$	5000	5000
$N_{\text{minibatch}}$	10	10
N_{seeds}	6	6
N_{test}	1000	1000
decay epsilon scheduling	0.001	0.001

Table 2: Algorithms parameters for `slot-filling`

Parameters	BFTQ	FTQ
architecture	256x128x64	128x64x32
regularisation	0.0005	0
activation	relu	relu
size beta encoder	50	-
initialisation	xavier	xavier
loss function	L2	L2
optimizer	adam	adam
learning rate	0.001	0.01
epoch (NN)	5000	400
normalize reward	true	true
epoch (FTQ)	15	15
$\widetilde{\mathscr{B}}$	0:0.01:1	-
γ	0.9	0.9
$N = \lvert\mathcal{D}\rvert$	10000	10000
$N_{\text{minibatch}}$	10	10
N_{seeds}	10	10
N_{test}	150	150
decay epsilon scheduling	0.0003	0.0003

Table 3: Algorithms parameters for `Highway-Env`

.3 Repoducibility

.3.1 Instructions for reproducibility

To reproduce the result displayed in Section 7.5, first install the following conventional libraries for python3: pycairo, numpy, scipy and pytorch. Then, on a Linux Operating System, execute the following commands:

```
# Install highway-env
pip3 install --user git+https://github.com/eleurent/rl-agents
# pull code
git clone https://github.com/ncarrara/budgeted-rl.git
# Change python path to the path of this repository
export PYTHONPATH="budgeted-rl/ncarrara"
# Navigate to budgeted-rl folder
cd budgeted-rl/ncarrara/budgeted_rl
# Run main script using any config file
# Choose the range of seeds you want to test on
python3 main/egreedy/main-egreedy.py config/slot-filling.json 0 6
python3 main/egreedy/main-egreedy.py config/corridors.json 0 4
python3 main/egreedy/main-egreedy.py config/highway-easy.json 0
    ↪ 10
```

Instructions to reproduce experiments

The GPU used for experiments is an NVIDIA `GeForce GTX 1080 Ti` and the CPU is an Intel `Xeon E7`.

.3.2 The Machine Learning reproducibility checklist

For all models and algorithms presented, indicate if you include:

- A clear description of the mathematical setting, algorithm, and/or model:
  - **yes**, see Section 7.1, Section 7.2, Section 7.3, and Section 7.4.
- An analysis of the complexity (time, space, sample size) of any algorithm:
  - **yes**, see Section 7.4.3.
- A link to a downloadable source code, with specification of all dependencies, including external libraries:
  - **yes**, see Appendix .3.1 and the folder `code` in the supplementary material zip file.

For any theoretical claim, indicate if you include:

- A statement of the result:
  - **yes**, see Section 7.2 and Section 7.3.
- A clear explanation of any assumptions:
  - we make one assumption in Section 7.2. We assume the program is feasible for any state. If not, no algorithm would be able to solve it anyway.
- A complete proof of the claim:
  - **yes**, see Appendix .1.

For all figures and tables that present empirical results, indicate if you include:

- A complete description of the data collection process, including sample size:
  - **yes**, see Section 7.5 and Appendix .2.
- A link to a downloadable version of the dataset or simulation environment:

  - **yes**, all environments are fetch from a public repository, see Appendix .3.1 for details.
- An explanation of any data that were excluded, description of any pre-processing step:
  - it is **not applicable** as data comes from simulated environments, so pre-processing steps are not needed.
- An explanation of how samples were allocated for training / validation / testing:
  - it is **not applicable**. The complete dataset is used for training. There is no need for validation set. Testing is performed in the true environment as in classical Online learning approaches.
- The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results:
  - **yes**, see Appendix .2.
- The exact number of evaluation runs:
  - **yes**, see N_{seeds} in the tables from Appendix .2.
- A description of how experiments were run:
- - **yes**, see the two first paragraphs of Section 7.5.
- A clear definition of the specific measure or statistics used to report results:
  - **yes**, see Section 7.5.2.
- Clearly defined error bars:
  - **yes**, we plot 95% confidence intervals in all figures, see Section 7.5.2.
- A description of results with central tendency (e.g. mean) variation (e.g. stddev):
  - **yes**, we even observe less variability with our novel approach, see Section 7.5.2.
- A description of the computing infrastructure used:
  - The GPU used for experiments is an NVIDIA `GeForce GTX 1080 Ti` and the CPU is an Intel `Xeon E7`.

A. Continuous transfer in Deep Q-learning

In this chapter, we include a transfer process inside the workflow of a Deep Q-learning (DQN) algorithm. As stated in Chapter 5, a Transfer Learning (TL) algorithm consists of two phases, a *transfer* phase where we choose what or how to transfer, and a *learning* phase where we learn the target policy using the transferred knowledge. There are different ways of implementing those phases in the DQN algorithm; we introduce some of them that failed in practice, and we explain why. First, we describe briefly how DQN works.

A.0.1 Deep Q-learning

DQN is an Online Reinforcement Learning (RL) algorithm fit for continuous state-space and discrete action-space. It uses a Neural Network (NN) as Q-function approximator. One can see DQN as an adaptation of the Q-learning algorithm using a NN as function approximator. Indeed, DQN updates the weights online with respect to the Temporal Difference (TD) error. In order to remember its previous interactions with the environment, DQN keeps the transitions in a memory called the Experience Replay. So, instead of considering a single interaction to update the weights, DQN draws a random set of previously encountered interactions[1], called the replay buffer, and computes the mean TD error over those interactions. To ensure stability, DQN is equipped with two networks:

- the *greedy* network, updated at each interaction, is used to compute the greedy action in order to interact with the environment.
- the *bootstrap* network[2], updated at a lower frequencies, is used to compute the argument of $\max$ operator in the TD error.

Now we identified the different components of DQN, we are able to incorporate transfer knowledge into it. Now let us assume that we are given a set of different users $\mathscr{U}$. For any

[1] In order to makes the data fed to the neural network independent and identically distributed (iid).

[2] In the literature, this network is called *target* network, but this nomenclature conflicts with the idea of target environment and target policy bought by the TL framework.

$u \in \mathscr{U}$, we know the optimal DP (and underlying greedy network) π_u and the batch of interactions $\mathcal{D}_u$ used to learn the DP.

A.1 The transfer phase.

First, to operate the *transfer* phase, we investigate two ideas to choose what source we transfer from:

A.1.1 Auto-Encoders

For each user u, we learn an Auto-Encoder AE_u that reconstruct the transitions contained in $\mathcal{D}_u$. The learning batch takes this form: $\{t_i, t_i\}_{i \in [0, |\mathcal{D}_u|[}$ where t is a concatenation of corresponding interaction's components: $t_i = (s_i, a_i, r'_i, s'_i)$. Then to select the closest source user to the target user, we forward the target interactions in each source Auto-Encoder and compute the test error. We postulated at first that the lowest test error would correspond to the closest source user. That statement showed promising results on a cart-pole environment and on PyDial. Unfortunately, the discriminative attribute that helped to select the best source was mostly the action. That makes sense when we think that the learning of the Auto-Encoder is actually biased by the action distribution in the batch, that is conditioned by the policy that generated this batch. Another limit of this approach is the fact that the Auto-Encoder may not really capture the transition function P as it may learn to reconstruct each s and s' independently if occurrences of any of those are high enough. One solution might be using the algorithm proposed in Ammar et al. (2014) based on Restricted Boltzmann Machines (RBMs), but they only encode (s_i, a_i, s'_i).

A.1.2 Using the Temporal Difference error

Here, we consider each source greedy Q-network. To select the closest source user, we compute the TD error of each source greedy network using the target batch of transitions. We made the hypothesis that the network that minimises this error may perform better with the target environment. This statement is false and we propose a simple counter example where the network with the lowest TD error is sub-optimal. Consider the simple target MDP in Figure A.1 where the starting state is s. As the MDP is deterministic and there are only two actions, we can represent a transition using its reward. So we assume the target batch $\mathcal{D}_{u_t} = \{1, 0\}$. Now let us say we have two source networks, Q_{u_0} and Q_{u_1}, and let $Q_{u_0}(a_0) = -1$, $Q_{u_0}(a_1) = 0$, $Q_{u_1}(a_0) = 5$ and $Q_{u_1}(a_1) = 0$. Then the average TD error of Q_{u_0} w.r.t $\mathcal{D}_{u_t}$ is 1 and the average TD error of Q_{u_1} is 2. So Q_{u_0} seems better than Q_{u_1}, however, the policy based on Q_{u_0} will always choose a_1 the sub-optimal action a_1 while the policy based on Q_{u_1} will always choose the best action a_0.

A.2 The learning phase

We also proposed two solutions for learning the policy through DQN using the transferred knowledge. Two questions arise, what data to transfer and when to stop the transfer.

A.2.1 Transferring the transition

In this solution, we tried to transfer the transitions of the best current source user (using the source selection of our choice, at this moment, we used the Auto-Encoder method). To

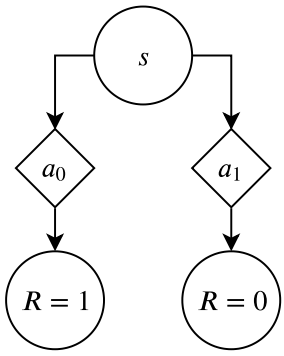


Figure A.1: Counter example Markov Decision Process of the Temporal Difference error solution

operate the transfer, we simply fill up the remaining space of the replay buffer with the source transitions. That way, the target DQN includes source knowledge into its learning. The advantage of this method is that once the replay buffer is full of target transitions, there is no more need to transfer source knowledge. Thus, there is no need to control the transfer through a meta parameter. The direct drawback of this, since replay buffer rarely exceeds 128 transitions, is that the target DQN prematurely stops using the source knowledge and the benefit of the transfer is not exploited enough. Another lead of research could be adding the transitions into the target experience replay directly, and clean it from source transitions at some point of the learning to avoid negative transfer. In this case, the question would be when to stop the transfer. Finally, all methods transfering the transitions face a major issue: at the very beginning of the learning, the policy is sub-optimal as it dit not have the time to bootstrap. Laroche, Trichelair, et al. (2019) showed recently that vanilla DQN is actually bad at batch learning. That led us to the next solution, that is transfering directly the Q-network.

A.2.2 Transferring the network

We tried two solutions for transferring the network.

The first one relies on a simple idea: using the network that minimise the TD error among all the source networks and the target network. As the target greedy network is learnt on the batch he is tested on, we must learnt an extra target network on a partial part of the batch, the learning base, and test it on the other part, the test base. The idea is similar to slice the learning batch in Supervised Learning (SL) problems. Unfortunatly, as we saw with the counterexample in Figure A.1, the TD is not an appropriate metric to evaluate what network is the best fit for the target environment.

The second solution consists on concatenating the current best source network (using the source selection of our choice) and the target network and weight their output in a single meta-network. The basic idea is to freeze the gradient of the source network and only learn the weights of the gate (that weights the output of the source and target network) and the weights of the greedy target. We also add extra information for the gate, as for example, the error return by the source Auto-Encoder. We believe that the meta-network should be able to emphasise the source network when the learning batch of the target environment

is small and emphasise the target network when it is strong enough to minimise the TD error. But once again, the foundation of the idea also relies on the fact that a small TD error should indicate a good fit for the target environment which is false.

A.3 Conclusion

The idea of transferring knowledge continuously into DQN is very tempting, convenient and does not need many changes to fit the initial framework. However, the Online nature of the algorithm makes it very hard to select the right source knowledge to transfer as the processus is not stationary. The two main challenges rely on what source to pick, and when to stop the transfer. We consider at the moment learning the R and P function using ideas from the model-based community (Deisenroth et al. 2011) in order to detect the closest source environment. The idea of combining model of the environments and a model-free algorithm may be counter-intuitive and unproductive, so a switch to a model-based algorithms may be appropriate.

A.3.1 Discussion

A lot of related work has been conducted since the beginning of this work. The closest of ours is Chaplot et al. (2016) where the authors transfer the features of the DQN model on DOOM 3d environment. To cite a few another recent works: Model-Agnostic Meta-Learning (Finn et al. 2017) for RL on MuJoCo; Successor Features - a generalisation of Successor Representations (Dayan 1993) - on the DeepMind Lab platform (Barreto et al. 2018), or on robot navigation tasks (J. Zhang et al. 2017); generalisation through simulation (Kang et al. 2019) for vision-based autonomous flight.

Bibliography

Abe, Naoki, Prem Melville, Cezar Pendus, Chandan K. Reddy, David L. Jensen, Vince P. Thomas, James J. Bennett, Gary F. Anderson, Brent R. Cooley, Melissa Kowalczyk, Mark Domick, and Timothy Gardinier (2010). "Optimizing Debt Collections Using Constrained Reinforcement Learning". In: *Conference of the Special Interest Group on Knowledge Discovery and Data Mining* (cited on page 86).

Achiam, Joshua, David Held, Aviv Tamar, and Pieter Abbeel (2017). "Constrained Policy Optimization". In: *International Conference on Machine Learning* (cited on page 86).

Allen, James (1995). *Natural Language Understanding (2Nd Ed.)* Benjamin-Cummings Publishing Co., Inc. (cited on page 20).

Altman, Eitan (1999). *Constrained Markov Decision Processes*. CRC Press (cited on page 70).

Ammar, Haitham Bou, Eric Eaton, Matthew E Taylor, Decebal Constantin Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls (2014). "An automated measure of mdp similarity for transfer in reinforcement learning". In: *Workshops at the Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 114).

Asimov, Isaac (1961). *The Machine That Won the War*. The Magazine of Fantasy and Science Fiction (cited on page 18).

— (1986). *Foundation and Earth*. Doubleday (cited on page 20).

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002a). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* (cited on page 41).

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002b). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* (cited on pages 55, 57).

Austin, John L. (1962). *How to do things with words*. Cambridge: Harvard University Press (cited on page 28).

Banach, Stefan (1922). "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". In: *Fundamenta Mathematicae* (cited on page 37).

Barlier, Merwan, Romain Laroche, and Olivier Pietquin (2018a). "Training Dialogue Systems With Human Advice". In: *International Conference on Autonomous Agents and Multiagent Systems* (cited on page 28).

— (2018b). "Training Dialogue Systems With Human Advice". In: *International Conference on Autonomous Agents and Multiagent Systems* (cited on page 40).

Barlier, Merwan, Julien Perolat, Romain Laroche, and Olivier Pietquin (2015a). "Human-Machine Dialogue as a Stochastic Game". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 55).

— (2015b). "Human-machine dialogue as a stochastic game". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 97).

Barreto, Andre, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos (2018). "Transfer in deep reinforcement learning using successor features and generalised policy improvement". In: (cited on page 116).

Bellman, Richard (1956). "Dynamic programming and Lagrange multipliers". In: *National Academy of Sciences of the USA* (cited on page 37).

— (1957). "A Markovian decision process". In: *Journal of Mathematics and Mechanics* (cited on pages 33, 37).

Bellman, Richard and S.E. Dreyfus (1959). "Functional Approximations and Dynamic Programming". In: *Mathematics of Computation* (cited on page 38).

Ben, Gold (1966). *Word Recognition Computer Program*. Technical report. Research Laboratory of Electronics. Cambrige; M. I. T. (cited on page 19).

Bertsekas, Dimitri P. (1996). *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific (cited on page 39).

Beutler, Frederick J. and Keith W. Ross (1985). "Optimal policies for controlled Markov chains with a constraint". In: *Journal of Mathematical Analysis and Applications* (cited on pages 70, 71, 86, 106).

Biermann, Alan W and Philip M Long (1996). "The composition of messages in speech-graphics interactive systems". In: *International Symposium on Spoken Dialogue* (cited on page 21).

Bińkowski, Mikołaj, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan (2019). "High Fidelity Speech Synthesis with Adversarial Networks". In: *arXiv:1909.11646* (cited on page 19).

Bobrow, Daniel G. (1964). *Natural Language Input for a Computer Problem Solving System*. Technical report (cited on page 20).

Bobrow, Daniel G., Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd (1977). "GUS, a Frame-driven Dialog System". In: *Artificial Intelligence* (cited on page 21).

Boutilier, Craig and Tyler Lu (2016). "Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes". In: *Conference on Uncertainty in Artificial Intelligence* (cited on pages 70, 86, 96).

Bruce, Wilcox (2011). *Rose*. http://brilligunderstanding.com/rosedemo.html (cited on page 22).

Budzianowski, Pawel and Ivan Vulic (2019). "Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems". In: *arxiv:1907.05774* (cited on page 97).

Carrara, Nicolas, Romain Laroche, Jean-Léon Bouraoui, Tanguy Urvoy, and Olivier Pietquin (2018a). "A Fitted-Q Algorithm for Budgeted MDPs". In: *Workshop on Safety, Risk and Uncertainty in Reinforcement Learning, Conference on Uncertainty in Artificial Intelligence* (cited on page 23).

— (2018b). "A Fitted-Q Algorithm for Budgeted MDPs". In: *European Workshop on Reinforcement Learning* (cited on page 23).

— (2018c). "Safe transfer learning for dialogue applications". In: *International Conference on Statistical Language and Speech Processing* (cited on pages 23, 50, 51).

Carrara, Nicolas, Romain Laroche, and Olivier Pietquin (2017). "Online learning and transfer for user adaptation in dialogue systems". In: *Joint special session on negotiation dialog, Workshop on the Semantics and Pragmatics of Dialogue- Conference of the Special Interest Group on Discourse and Dialogue* (cited on pages 23, 50, 51).

Carrara, Nicolas, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Jean-Léon Bouraoui, Odalric Maillard, and Olivier Pietquin (2019). "Budgeted Reinforcement Learning in Continuous State Space". In: *Workshop on Safety Risk and Uncertainty in Reinforcement Learning at Conference on Uncertainty in Artificial Intelligence (2018), European Workshop on Reinforcement Learning (2018), and Conference on Neural Information Processing Systems (2019)* (cited on page 23).

Carrara, Nicolas, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric Maillard, and Olivier Pietquin (2019). "Budgeted Reinforcement Learning in Continuous State Space". In: (cited on page 23).

Casanueva, Inigo, Thomas Hain, Heidi Christensen, Ricard Marxer, and Phil Green (2015a). "Knowledge transfer between speakers for personalised dialogue management". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on pages 49–51).

— (2015b). " Knowledge transfer between speakers for personalised dialogue management ". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 89).

Chandramohan, Senthilkumar, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin (2012). " Clustering behaviors of spoken dialogue systems users ". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on page 55).

— (2014). "Co-adaptation in spoken dialogue systems". In: *Natural interaction with robots, knowbots and smartphones*. Springer (cited on page 97).

Chandramohan, Senthilkumar, Matthieu Geist, and Olivier Pietquin (2010). "Optimizing Spoken Dialogue Management with Fitted Value Iteration". In: *Conference of the International Speech Communication Association* (cited on pages 55, 80, 89).

Chaplot, Devendra Singh, Guillaume Lample, Kanthashree Mysore Sathyendra, and Ruslan Salakhutdinov (2016). "Transfer deep reinforcement learning in 3d environments: An empirical study". In: *Deep Reinforcement Learning Workshop at Conference on Neural Information Processing Systems* (cited on page 116).

Chen, Hongshen, Xiaorui Liu, Dawei Yin, and Jiliang Tang (2017). "A Survey on Dialogue Systems: Recent Advances and New Frontiers". In: *Exploration Newsletter* (cited on pages 22, 27).

Chen, Lu, Cheng Chang, Zhi Chen, Bowen Tan, Milica Gasic, and Kai Yu (2018). " Policy Adaptation for Deep Reinforcement Learning-Based Dialogue Management ". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on pages 48, 51).

Chorowski, Jan and Navdeep Jaitly (2017). "Towards better decoding and language model integration in sequence to sequence models". In: *Conference of the International Speech Communication Association* (cited on page 27).

Chow, Yinlam, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone (2018). "Risk-Constrained Reinforcement Learning with Percentile Risk Criteria". In: *Journal of Machine Learning Research* (cited on pages 70, 86).

Chow, Yinlam, Aviv Tamar, Shie Mannor, and Marco Pavone (2015). " Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach ". In: *Conference on Neural Information Processing Systems* (cited on page 70).

Clarke, Arthur Charles (1968). *2001: A Space Odyssey*. Hutchinson (cited on page 18).

Colby, Kenneth Mark (1975). *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc. (cited on page 21).

Cortada, James W. (2005). *The Digital Hand: Volume II: How Computers Changed the Work of American Financial, Telecommunications, Media, and Entertainment Industries*. Oxford University Press (cited on page 21).

Cuayahuitl, Heriberto, Simon Keizer, and Oliver Lemon (2015). "Strategic Dialogue Management via Deep Reinforcement Learning". In: *Workshop on Deep Reinforcement Learning, Conference on Neural Information Processing Systems* (cited on page 43).

Cummings, Louise (2010). *The Routledge pragmatics encyclopedia*. Routledge (cited on page 21).

Dann, Christoph, Lihong Li, Wei Wei, and Emma Brunskill (2019). "Policy Certificates: Towards Accountable Reinforcement Learning". In: *International Conference on Machine Learning* (cited on page 70).

Davis, KH, R Biddulph, and Stephen Balashek (1952). "Automatic recognition of spoken digits". In: *Journal of the Acoustical Society of America* (cited on page 19).

Dayan, Peter (1993). "Improving generalization for temporal difference learning: The successor representation". In: *Neural Computation* (cited on page 116).

Deisenroth, Marc Peter and Carl Edward Rasmussen (2011). " PILCO: A Model-Based and Data-Efficient Approach to Policy Search ". In: *International Conference on Machine Learning* (cited on page 116).

Deng, Li, Gökhan Tür, Xiaodong He, and Dilek Z. Hakkani-Tür (2012). "Use of kernel deep convex networks and end-to-end learning for spoken language understanding." In: *IEEE Spoken Language Technology Workshop* (cited on page 28).

Dennis, Klatt (1987). "How Klattalk became DECtalk: An Academic's Experiences in the Business World". In: *The official proceedings of Speech Technology, New York* (cited on page 19).

Deoras, Anoop and Ruhi Sarikaya (2013). "Deep belief network based semantic taggers for spoken language understanding." In: *Conference of the International Speech Communication Association* (cited on page 28).

Eckert, Wieland, Esther Levin, and Roberto Pieraccini (1997). "User modeling for spoken dialogue system evaluation". In: *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings* (cited on page 49).

Ehsan, Upol, Brent Harrison, Larry Chan, and Mark O. Riedl (2018). "Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations". In: *Conference on AI, Ethics, and Society, Association for Computing Machinery-Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 19).

El Asri, Layla (2016). "Learning the Parameters of Reinforcement Learning from Data for Adaptive Spoken Dialogue Systems". PhD thesis. Université de Lorraine (cited on page 34).

El Asri, Layla, Jing He, and Kaheer Suleman (2016). "A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems". In: *Conference of the International Speech Communication Association* (cited on page 49).

El Asri, Layla, Hatim Khouzaimi, Romain Laroche, and Olivier Pietquin (2014). "Ordinal regression for interaction quality prediction". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on page 42).

El Asri, Layla and Romain Laroche (2013). "Will my Spoken Dialogue System be a Slow Learner?" In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 42).

El Asri, Layla, Romain Laroche, and Olivier Pietquin (2012). "Reward Function Learning for Dialogue Management." In: *Frontiers in Artificial Intelligence and Applications* (cited on page 51).

— (2014). " Task Completion Transfer Learning for Reward Inference ". In: *Workshop, Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on pages 50, 51).

Engel, Yaakov, Shie Mannor, and Ron Meir (2006). " Reinforcement learning with Gaussian processes ". In: *International Conference on Machine Learning* (cited on page 48).

Ernst, Damien, Pierre Geurts, and Louis Wehenkel (2005). "Tree-Based Batch Mode Reinforcement Learning". In: *Journal of Machine Learning Research* (cited on pages 40, 55).

Farahmand, Amir massoud, Mohammad Ghavamzadeh, Csaba Szepesvari, and Shie Mannor (2009). "Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems". In: (cited on page 64).

Fedus, William, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle (2019). "Hyperbolic discounting and learning over multiple horizons". In: *arXiv preprint arXiv:1902.06865* (cited on page 36).

Ferguson, Kimberly and Sridhar Mahadevan (2006). "Proto-transfer learning in markov decision processes using spectral methods". In: *Computer Science Department Faculty Publication Series* (cited on page 46).

Ferrante, Eliseo, Alessandro Lazaric, and Marcello Restelli (2008). "Transfer of task representation in reinforcement learning using policy-based proto-value functions". In: *International Conference on Autonomous Agents and Multiagent Systems* (cited on page 46).

Ferreira, Emmanuel and Fabrice Lefèvre (2013). "Social Signal and User Adaptation in Reinforcement Learning-based Dialogue Management". In: *Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication* (cited on page 41).

Ferrucci, David, Eric Brown, Jennifer Chu-carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, and John Prager (2010). "Building Watson: An Overview of the DeepQA Project". In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 20).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning* (cited on page 116).

Fukushima, Kunihiko (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36 (cited on page 28).

Gao, Jianfeng, Michel Galley, Lihong Li, et al. (2019). "Neural approaches to conversational AI". In: *Foundations and Trends ® in Information Retrieval* (cited on pages 21, 30).

Garcia, Javier and Fernando Fernandez (2015). " A Comprehensive Survey on Safe Reinforcement Learning ". In: *Journal of Machine Learning Research (JMLR)* (cited on page 70).

Gasic, Milica, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young (2013). "POMDP-based dialogue manager adaptation to extended domains". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on pages 48–51).

Gasic, Milica and Steve Young (2013). "Gaussian processes for pomdp-based dialogue manager optimization". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (cited on page 48).

Geibel, Peter and Fritz Wysotzki (2005). "Risk-sensitive reinforcement learning applied to control under constraints." In: *Journal of Artificial Intelligence Research* (cited on page 86).

Genevay, Aude and Romain Laroche (2016). "Transfer Learning for User Adaptation in Spoken Dialogue Systems". In: *International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems (cited on pages 23, 50, 51, 55–57, 63, 89, 95).

Georgila, Kallirroi and David R Traum (2011). "Reinforcement Learning of Argumentation Dialogue Policies in Negotiation." In: *Conference of the International Speech Communication Association* (cited on page 55).

Glass, James, Giovanni Flammia, David Goodine, Michael Phillips, Joseph Polifroni, Shinsuke Sakai, Stephanie Seneff, and Victor Zue (1995). "Multilingual spoken-language understanding in the MIT Voyager system". In: *Speech Communication* (cited on page 21).

Goddeau, David, Helen M. Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai (1996). "A form-based dialogue manager for spoken language applications". In: *International Conference on Spoken Language Processing* (cited on page 28).

Goldberg, Eli, Norbert Driedger, and Richard I. Kittredge (1994). "Using Natural-Language Processing to Produce Weather Forecasts". In: *IEEE Expert: Intelligent Systems and Their Applications* (cited on page 19).

Graves, Alex and Navdeep Jaitly (2014). "Towards End-to-end Speech Recognition with Recurrent Neural Networks". In: *International Conference on Machine Learning* (cited on pages 19, 27).

Hashemi, Homa Baradaran (2016). "Query Intent Detection using Convolutional Neural Networks". In: *Workshop on Query Understanding, International Conference on Web Search and Data Mining* (cited on page 28).

Henderson, Matthew (2015). "Machine Learning for Dialog State Tracking: A Review". In: *International Workshop on Machine Learning in Spoken Language Processing* (cited on page 28).

Henderson, Matthew, Blaise Thomson, and Steve Young (2013). "Deep Neural Network Approach for the Dialog State Tracking Challenge". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 21).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-term Memory". In: *Neural computation* (cited on page 19).

Homer, Dudley, Riesz R., and Watkins S. (1939). "A Synthetic Speaker". In: *Journal of Franklin Institute, Philadelphia* (cited on page 18).

Howard, Ronald A. (1960). *Dynamic Programming and Markov Processes*. MIT Press (cited on page 37).

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck (2013). "Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data". In: *ACM International Conference on Information & Knowledge Management* (cited on page 28).

Ilievski, Vladimir, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl (2018). "Goal-Oriented Chatbot Dialog Management Bootstrapping with Transfer Learning". In: *International Joint Conference on Artificial Intelligence* (cited on pages 48, 51).

Isaac, Oscar (2015). *Ex machina* (cited on page 20).

Iyengar, Garud N. (2005). " Robust Dynamic Programming ". In: *Mathematics of Operations Research* (cited on page 70).

J. C. H. Watkins, Christopher and Peter Dayan (1992). "Q-learning". In: *Machine Learning* (cited on page 89).

Janarthanam, Srinivasan and Oliver Lemon (2010). "Adaptive referring expression generation in spoken dialogue systems: Evaluation with real users". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 55).

Jelinek, Frederick (1976). "Continuous Speech Recognition by Statistical Methods". In: *IEEE 64* (cited on page 19).

Jonze, Spike (2013). *Her* (cited on page 21).

Jurafsky, Daniel and James H. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR (cited on page 27).

Kang, Katie, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine (2019). "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight". In: *arXiv preprint arXiv:1902.03701* (cited on pages 97, 116).

Karl Johan, Astrom (1965). "Optimal control of Markov processes with incomplete state information". In: *Journal of Mathematical Analysis and Applications* (cited on page 21).

Kaufmann, Leonard and Peter Rousseeuw (1987). "Clustering by Means of Medoids". In: *Data Analysis based on the L1-Norm and Related Methods* (cited on page 55).

Keizer, Simon, Milica Gasic, Filip Jurcicek, Francois Mairesse, Blaise Thomson, Kai Yu, and Steve Young (2010). "Parameter estimation for agenda-based user simulation". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 49).

Keizer, Simon and Verena Rieser (2016). " The MaDrIgAL Project: Multi-Dimensional Interaction Management and Adaptive Learning ". In: *International Workshop on Domain Adaptation for Dialog Agents* (cited on page 48).

— (2018). " Towards Learning Transferable Conversational Skills using Multi-dimensional Dialogue Modelling ". In: *Workshop on the Semantics and Pragmatics of Dialogue* (cited on pages 48, 51).

Khouzaimi, Hatim, Romain Laroche, and Fabrice Lefevre (2015). " Optimising turn-taking strategies with reinforcement learning. " In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 58).

— (2017). "Incremental human-machine dialogue simulation". In: *Dialogues with Social Robots*. Springer (cited on page 49).

Kim, Jaeyoung, Mostafa El-Khamy, and Jungwon Lee (2017). "Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition". In: *Conference of the International Speech Communication Association* (cited on page 27).

Kim, Suyoun and Michael L. Seltzer (2018). "Towards Language-Universal End-to-End Speech Recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on page 27).

Lagoudakis, Michail G. and Ronald Parr (2003). "Least-squares Policy Iteration". In: *Journal of Machine Learning Research* (cited on page 40).

Langley, Pat (2006). "Transfer of knowledge in cognitive systems". In: *workshop on Structural Knowledge Transfer for Machine Learning at International Conference on Machine Learning* (cited on page 47).

Laroche, Romain (2017). "The complex negotiation dialogue game". In: *Workshop on the Semantics and Pragmatics of Dialogue* (cited on pages 49, 96).

Laroche, Romain, Philippe Bretier, and Ghislain Putois (2010). "Enhanced monitoring tools and online dialogue optimisation merged into a new spoken dialogue system design experience". In: *Conference of the International Speech Communication Association* (cited on page 42).

Laroche, Romain and Aude Genevay (2017). "The negotiation dialogue game". In: *Dialogues with Social Robots*. Springer (cited on pages 23, 55, 58).

Laroche, Romain, Ghislain Putois, and Philippe Bretier (2010). "Optimising a handcrafted dialogue system design". In: *Conference of the International Speech Communication Association* (cited on page 42).

Laroche, Romain, Ghislain Putois, Philippe Bretier, and Bernadette Bouchon-Meunier (2009). "Hybridisation of expertise and reinforcement learning in dialogue systems". In: *Conference of the International Speech Communication Association* (cited on page 42).

Laroche, Romain, Paul Trichelair, and Remi Tachet des Combes (2019). "Safe Policy Improvement with Baseline Bootstrapping". In: (cited on pages 86, 115).

Larson, Glen Albert (1986). *Knight Rider*. National Broadcasting Company (cited on page 18).

Lavoie, Benoit, Owen Ranbow, and Ehud Reiter (1977). "Customizable Descriptions of Object-Oriented Models." In: *Advances in Natural Language Processing* (cited on page 19).

Lazaric, Alessandro (2008). "Knowledge transfer in reinforcement learning". PhD thesis. Poltecnico di Milano (cited on page 46).

— (2012). "Transfer in Reinforcement Learning: a Framework and a Survey". In: *Reinforcement Learning - State of the art*. Edited by Martijn van Otterlo Marco Wiering. Springer (cited on pages 45–47, 89).

Lazaric, Alessandro, Marcello Restelli, and Andrea Bonarini (2008). "Transfer of samples in batch reinforcement learning". In: *International Conference on Machine Learning* (cited on pages 55, 57).

Le, Hoang M., Cameron Voloshin, and Yisong Yue (2019). "Batch Policy Learning under Constraints". In: *International Conference on Machine Learning* (cited on page 86).

LeCun, Yann, Patrick Haffner, L é on Bottou, and Yoshua Bengio (1999). "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision* (cited on page 28).

Lee, Kyungmin, Chiyoun Park, Namhoon Kim, and Jaewon Lee (2018). "Accelerating Recurrent Neural Network Language Model Based Online Speech Recognition System". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on page 27).

Lemon, Oliver (2011). "Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation". In: *Computer Speech & Language* (cited on page 19).

Lemon, Oliver and Xingkun Liu (2007). "Dialogue policy learning for combinations of noise and user simulation: transfer results". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 49).

Lemon, Oliver and Olivier Pietquin (2012). *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. Springer Science & Business Media (cited on page 21).

Leurent, Edouard, Yann Blanco, Denis Efimov, and Odalric-Ambrym Maillard (2018). " Approximate Robust Control of Uncertain Dynamical Systems ". In: *Workshop on Machine Learning for Intelligent Transportation Systems, Conference on Neural Information Processing Systems* (cited on page 81).

Levin, Esther and Roberto Pieraccini (1997). "A stochastic model of computer-human interaction for learning dialogue strategies." In: *European Conference on Speech Communication and Technology* (cited on pages 21, 49).

Levin, Esther, Roberto Pieraccini, and Wieland Eckert (2000). "A stochastic model of human-machine interaction for learning dialog strategies". In: *IEEE Transactions on speech and audio processing* (cited on page 49).

Li, Jiwei, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao (2016). "Deep Reinforcement Learning for Dialogue Generation". In: *Conference on Empirical Methods in Natural Language Processing* (cited on page 29).

Li, Lihong, Jason D Williams, and Suhrid Balakrishnan (2009). "Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection". In: *International Speech Communication Association* (cited on page 40).

Li, Lihong, Jason Williams, and Suhrid Balakrishnan (2009). "Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection". In: *Conference of the International Speech Communication Association* (cited on pages 55, 80).

Li, Xiujun, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz (2017). "End-to-end task-completion neural dialogue systems". In: (cited on page 97).

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Workshop on Text Summarization Branches Out, Annual Meeting of the Association for Computational Linguistics* (cited on page 20).

Lison, Pierre (2013). "Model-based Bayesian Reinforcement Learning for Dialogue Management". In: *Conference of the International Speech Communication Association* (cited on page 40).

Liu, Chunming, Xin Xu, and Dewen Hu (2014). " Multiobjective Reinforcement Learning: A Comprehensive Overview ". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (cited on page 70).

Lowe, Ryan, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau (2017). "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses". In: *Annual Meeting of the Association for Computational Linguistics* (cited on page 20).

Lowe, Ryan, Nissan Pow, Iulian Serban, and Joelle Pineau (2015). *The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems*. Technical report (cited on page 39).

Luenberger, David G. (2013). *Investment science*. Oxford University Press, Incorporated (cited on page 70).

MacQueen, James (1967). "Some methods for classification and analysis of multivariate observations". In: *Berkeley symposium on mathematical statistics and probability* (cited on page 55).

Mahadevan, Sridhar and Mauro Maggioni (2007). "Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes". In: *Journal of Machine Learning Research* (cited on page 46).

Mahmud, MM, Majd Hawasly, Benjamin Rosman, and Subramanian Ramamoorthy (2013). *Clustering markov decision processes for continual transfer*. Technical report (cited on pages 55, 65).

Mausser, Helmut and Dan Rosen (2003). "Beyond VaR: from measuring risk to managing risk". In: *IEEE Conference on Computational Intelligence for Financial Engineering* (cited on page 70).

Mesnil, Gregoire, Xiaodong He, Li Deng, and Yoshua Bengio (2013). "Investigation of recurrent-neural-network architectures and learning methods for spoken language

understanding." In: *Conference of the International Speech Communication Association* (cited on page 28).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* (cited on pages 43, 77).

Mo, Kaixiang, Shuangyin Li, Yu Zhang, Jiajun Li, and Qiang Yang (2018). " Personalizing a Dialogue System with Transfer Reinforcement Learning ". In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on pages 50, 51).

Moore, Graham (2014). *Imitation Game* (cited on page 20).

Nadjahi, Kimia, Romain Laroche, and Remi Tachet des Combes (2019). "Safe Policy Improvement with Soft Baseline Bootstrapping". In: *European Conference on Machine Learning* (cited on page 86).

Neustein, Amy and Judith A Markowitz (2013). *Mobile speech and advanced natural language solutions*. Springer Science & Business Media (cited on page 22).

Nilim, Arnab and Laurent El Ghaoui (2005). " Robust Control of Markov Decision Processes with Uncertain Transition Matrices ". In: *Mathematics of Operations Research* (cited on page 70).

Oh, Alice H and Alexander I Rudnicky (2000). "Stochastic language generation for spoken dialogue systems". In: *Workshop on Conversational Systems, Advances in Natural Language Processing -NAACL* (cited on page 19).

Oord, Aäron van den, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis (2018). "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *International Conference on Machine Learning* (cited on page 29).

OpenAI (2018). *OpenAI Five*. https://blog.openai.com/openai-five/ (cited on page 96).

Palossi, Daniele, Francesco Conti, and Luca Benini (2019). "An Open Source and Open Hardware Deep Learning-powered Visual Navigation Engine for Autonomous Nano-UAVs". In: *arxiv:1905.04166* (cited on page 97).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Annual Meeting of the Association for Computational Linguistics* (cited on page 20).

Peng, Baolin, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su (2018). "Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning". In: *Annual Meeting of the Association for Computational Linguistics* (cited on page 40).

Perera, Rivindu and Parma Nand (2017). "Recent Advances in Natural Language Generation: A Survey and Classification of the Empirical Literature". In: *Computing and Informatics* (cited on page 19).

Perez, Sarah (2016). *Microsoft silences its new A.I. bot Tay, after Twitter users teach it racism*. URL: https://techcrunch.com/2016/03/24/microsoft-silences-its-new-a-i-bot-tay-after-twitter-users-teach-it-racism/ (visited on 2016) (cited on page 97).

Petrik, Marek, Mohammad Ghavamzadeh, and Yinlam Chow (2016). "Safe policy improvement by minimizing robust baseline regret". In: *Conference on Neural Information Processing Systems* (cited on page 86).

Pietquin, Olivier (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*. Presses Universitaires de Louvain (cited on pages 21, 49).

Pietquin, Olivier, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet (2011). "Sample-efficient batch reinforcement learning for dialogue management optimization". In: *ACM Transactions on Speech and Language Processing (TSLP)* (cited on pages 40, 55, 80).

Poupart, Pascal, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling (2015). "Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes". In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 86).

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners". In: (cited on pages 22, 97).

Rambow, Owen, Srinivas Bangalore, and Marilyn Walker (2001). "Natural Language Generation in Dialog Systems". In: *International Conference on Human Language Technology Research* (cited on page 19).

Rasmussen, Carl Edward (2003). *Gaussian processes in machine learning* (cited on page 48).

Riedmiller, Martin (2005). "Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method". In: *European Conference on Machine Learning* (cited on pages 40, 42, 77).

Rogers, Carl (1942). "Counseling and psychotherapy". In: *Cambridge, MA: Riverside Press* (cited on page 21).

Roijers, Diederik M., Peter Vamplew, Shimon Whiteson, and Richard Dazeley (2013). "A Survey of Multi-Objective Sequential Decision-Making". In: *Journal of Artificial Intelligence Research* (cited on pages 70, 71).

Rosenblatt, Frank (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (cited on page 19).

Roy, Nicholas, Joelle Pineau, and Sebastian Thrun (2000). "Spoken dialogue management using probabilistic reasoning". In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (cited on page 21).

Rubin, Philip and Louis Goldstein (2019). *The Pattern Playback*. URL: http://www.haskins.yale.edu/featured/patplay.html (visited on 06/08/2019) (cited on page 18).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: MIT Press (cited on page 19).

Rummery, Gavin A and Mahesan Niranjan (1994). *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England (cited on page 50).

Sadek, M David, Philippe Bretier, and Franck Panaget (1997). "ARTIMIS: Natural dialogue meets rational agency". In: (cited on pages 21, 28).

Sadri, Fariba, Francesca Toni, and Paolo Torroni (2001). "Dialogues for negotiation: agent varieties and dialogue sequences". In: *International Workshop on Agent Theories, Architectures, and Languages* (cited on page 55).

Saeed, Hira (2019). *RightClick.io Uses AI-Powered Chatbot to Create a Website*. URL: https://chatbotsmagazine.com/rightclick-io-ai-website-builder-82f0e6f3c61c (visited on 06/03/2019) (cited on page 22).

Sarikaya, Ruhi, Geoffrey E. Hinton, and Bhuvana Ramabhadran (2011). "Deep belief nets for natural language call-routing". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cited on page 28).

Schank, Roger C. and Larry Tesler (1969). "A Conceptual Dependency Parser for Natural Language". In: *Conference on Computational Linguistics* (cited on page 20).

Schatzmann, Jost (2008). *Statistical User and Error Modelling for Spoken Dialogue Systems*. University of Cambridge (cited on page 49).

Schatztnann, Jost, Matthew N Stuttle, Karl Weilhammer, and Steve Young (2005). "Effects of the user model on simulation-based learning of dialogue strategies". In: *IEEE Workshop on Automatic Speech Recognition and Understanding* (cited on page 49).

Scott, Ridley (1982). *Bladerunner* (cited on page 20).

Searle, John R. (1969). *Speech acts: An essay in the philosophy of language*. Cambridge university press (cited on page 28).

Serban, Iulian Vlad, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau (2015). *A survey of available corpora for building data-driven dialogue systems*. Technical report (cited on page 40).

Serban, Iulian Vlad, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau (2016). "Building end-to-end dialogue systems using generative hierarchical neural network models". In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on pages 30, 97).

Shen, Yelong, Xiaodong He, Jianfeng Gao, Li Deng, and Gr é goire Mesnil (2014). "Learning Semantic Representations Using Convolutional Neural Networks for Web Search". In: *International Conference on World Wide Web* (cited on page 28).

Sherstov, Alexander A and Peter Stone (2005). "Improving action selection in MDP's via knowledge transfer". In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 46).

Silver, David, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* (cited on page 96).

Singh, Satinder, Diane Litman, Michael Kearns, and Marilyn Walker (2002). "Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system". In: *Journal of Artificial Intelligence Research* (cited on page 42).

Steinhaus, Hugo (1957). "Sur la division des corps matériels en partie". In: *Bulletin de l'academie polonaise des sciences* (cited on page 55).

Sunmola, Funlade T and Jeremy L Wyatt (2006). "Model transfer for Markov decision tasks via parameter matching". In: *Workshop of the UK Planning and Scheduling Special Interest Group* (cited on page 46).

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* (cited on page 46).

Tamar, Aviv, Dotan Di Castro, and Shie Mannor (2012). " Policy Gradients with Variance Related Risk Criteria ". In: *International Conference on Machine Learning* (cited on page 70).

Taylor, Matthew E, Nicholas K Jong, and Peter Stone (2008). "Transferring instances for model-based reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (cited on page 46).

Taylor, Matthew E and Peter Stone (2009). "Transfer learning for reinforcement learning domains: A survey". In: *Journal of Machine Learning Research* (cited on pages 45, 89).

Taylor, Matthew E, Shimon Whiteson, and Peter Stone (2007). "Transfer via inter-task mappings in policy search reinforcement learning". In: *International Conference on Autonomous Agents and Multiagent Systems* (cited on page 46).

Terry, Hugh (2019). *Baidu's Melody – AI Powered Conversational Bot for Doctors and Patients - The Digital Insurer*. URL: https://www.the-digital-insurer.com/dia/baidus-melody-ai-powered-conversational-bot-for-doctors-and-patients-1/ (visited on 06/03/2019) (cited on page 22).

Thomas, Philip, Georgios Theocharous, and Mohammad Ghavamzadeh (2015). "High confidence policy improvement". In: *International Conference on Machine Learning* (cited on page 86).

Thomson, Blaise (2013). *Statistical Methods for Spoken Dialogue Management*. Springer Publishing Company, Incorporated (cited on page 28).

Thomson, Blaise and Steve Young (2010). "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems". In: *Computer Speech & Language* (cited on page 48).

Tikhonov, Andrei Nikolaevich (1963). "Regularization of incorrectly posed problems". In: *Doklady Akademii Nauk SSSR* (cited on page 64).

Torrey, Lisa, Trevor Walker, Jude Shavlik, and Richard Maclin (2005). "Using advice to transfer knowledge acquired in one reinforcement learning task to another". In: *European Conference on Machine Learning* (cited on page 46).

Tür, Gökhan, Li Deng, Dilek Z. Hakkani-Tür, and Xiaodong He (2012). "Towards deeper understanding: Deep convex networks for semantic utterance classification". In:

IEEE International Conference on Acoustics, Speech and Signal Processing (cited on page 28).

Turing, Alan Madison (1936). "On Computable Numbers, with an Application to the Entscheidungs problem". In: *London Mathematical Society* (cited on page 20).

— (1950). "Computing machinery and intelligence". In: *Mind* (cited on page 20).

Ultes, Stefan, Matthias Kraus, Alexander Schmitt, and Wolfgang Minker (2015). "Quality-adaptive spoken dialogue initiative selection and implications on reward modelling". In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on page 55).

Ultes, Stefan, Lina Maria Rojas-Barahona, Pei-hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Pawel Budzianowski, Nikola Mrksic, Tsung-Hsien Wen, Milica Gasic, and Steve J. Young (2017). "PyDial: A Multi-domain Statistical Dialogue System Toolkit". In: *Annual Meeting of the Association for Computational Linguistics* (cited on page 22).

Undurti, Aditya, Alborz Geramifard, Nicholas Roy, and Jonathan P How (2010). *Function Approximation for Continuous Constrained MDPs*. Technical report (cited on page 86).

Van Den Oord, A ä ron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu (2016). "WaveNet: A generative model for raw audio." In: *Speech Synthesis Workshop* (cited on pages 19, 29).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Conference on Neural Information Processing Systems* (cited on page 22).

Vinyals, Oriol, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver (2019). *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. URL: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii (visited on 2019) (cited on page 97).

Vries, Harm de, Florian Strub, A. P. Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville (2017). "GuessWhat?! Visual Object Discovery through Multi-modal Dialogue". In: *IEEE Conference on Computer Vision and Pattern Recognition* (cited on pages 29, 43).

W. Forgie, James and Carma D. Forgie (1959). "Results Obtained from a Vowel Recognition Computer Program". In: *The Journal of the Acoustical Society of America* (cited on page 19).

Walker, Marilyn A (1993). "Informational redundancy and resource bounds in dialogue". PhD thesis. University of Pennsylvania, The Institute for Research in Cognitive Science (cited on page 21).

Walker, Marilyn A, Jeanne C Fromer, and Shrikanth Narayanan (1998). "Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email". In: *Annual Meeting of the Association for Computational Linguistics* (cited on page 21).

Wang, Yuxuan, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Robert Clark, and Rif A. Saurous (2017). "Tacotron: Towards End-to-End Speech Synthesis". In: *Conference of the International Speech Communication Association* (cited on page 29).

Wang, Ziyu, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas (2016). "Sample efficient actor-critic with experience replay". In: *arXiv preprint arXiv:1611.01224* (cited on page 43).

Weisz, Gellert, Pawel Budzianowski, Pei-Hao Su, and Milica Gasic (2018). "Sample Efficient Deep Reinforcement Learning for Dialogue Systems With Large Action Spaces". In: *IEEE/ACM Transactions Audio, Speech and Language Processing* (cited on page 43).

Weizenbaum, Joseph (1966). "Eliza—a computer program for the study of natural language com- munication between man and machine." In: *Communications of the Association for Computing Machinery* (cited on page 21).

Wen, Tsung - Hsien, Yishu Miao, Phil Blunsom, and Steve J. Young (2017). "Latent Intention Dialogue Models". In: *International Conference on Machine Learning* (cited on page 28).

Wen, Tsung-Hsien, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young (2015). *Semantically conditioned lstm-based natural language generation for spoken dialogue systems*. Technical report (cited on page 29).

Weng, John J., Narendra Ahuja, and Thomas S. Huang (1993). "Learning recognition and segmentation of 3-D objects from 2-D images". In: *International Conference on Computer Vision* (cited on page 28).

Wiesemann, Wolfram, Daniel Kuhn, and Berç Rustem (2013). "Robust Markov Decision Processes". In: *Mathematics of Operations Research* (cited on page 70).

Williams, Jason D, Pascal Poupart, and Steve Young (2008). “Partially observable Markov decision processes with continuous observations for dialogue management”. In: *Recent Trends in Discourse and Dialogue*. Springer (cited on page 42).

Williams, Jason D., Antoine Raux, Deepak Ramachandran, and Alan W. Black (2013). “The Dialog State Tracking Challenge”. In: *Conference of the Special Interest Group on Discourse and Dialogue* (cited on pages 21, 29, 40).

Williams, Ronald J. (1992). “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* (cited on page 43).

Worswick, Steve (2005). *Mitsuku*. https://www.pandorabots.com/mitsuku/ (cited on page 22).

Yan, Rui (2018). “"Chitty-Chitty-Chat Bot": Deep Learning for Conversational AI”. In: *International Joint Conference on Artificial Intelligence* (cited on page 30).

Yan, Zhao, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li (2017). “Building Task-Oriented Dialogue Systems for Online Shopping”. In: *Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence* (cited on page 22).

Yann, D, G Tur, D Hakkani-Tur, and L Heck (2014). “Zero-shot learning and clustering for semantic utterance classification using deep learning”. In: International Conference on Learning Representations (cited on page 28).

Yao, Kaisheng, Baolin Peng, Shuyuan Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi (2014). “Spoken language understanding using long short-term memory neural networks”. In: *IEEE Spoken Language Technology Workshop* (cited on page 28).

Yao, Kaisheng, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu (2013). “Recurrent neural networks for language understanding”. In: *Conference of the International Speech Communication Association* (cited on page 28).

Young, Steve J., Milica Gasic, Blaise Thomson, and Jason D. Williams (2013). “POMDP-Based Statistical Spoken Dialog Systems: A Review”. In: *IEEE* (cited on page 21).

Young, Steve, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu (2009). “The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management”. In: *Computer Speech and Language* (cited on page 29).

Zhang, Jingwei, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard (2017). “Deep reinforcement learning with successor features for navigation across similar environments”. In: *International Conference on Intelligent Robots and Systems*. IEEE (cited on page 116).

Zhang, Xiaodong and Houfeng Wang (2016). "A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding". In: *International Joint Conference on Artificial Intelligence* (cited on page 28).

Zue, Victor, Stephanie Seneff, James Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington (2000). "Jupiter: A Telephone-Based Conversational Interface for Weather Information". In: *IEEE Transactions on Speech and Audio Processing* (cited on page 21).

Index

Symbols

ε-greedy 41, 42, 57, 61, 65, 75, 76, 78, 96

A

adaptation 48–51
agent 17–21, 34, 36, 39, 41, 45, 46, 48, 81, 89, 90, 92, 96, 108

B

batch . . . 39, 40, 42, 55, 57, 58, 61, 64, 65, 71, 75, 76, 78, 80, 82, 86, 96, 114, 115
behavioural policy 41
Bellman Evaluation 36, 40, 72
Bellman Optimality 37, 38, 40, 69, 71
budget 69–72, 96, 107
budgeted policy 71–73, 75, 82, 96

C

clustering 24, 50, 55, 56, 59, 65

D

deterministic 114
deterministic policy 36, 71, 86, 106
dialogue 19–23, 28–30, 33–36, 38–41, 45, 48–51, 55–61, 63–66, 69, 80, 82, 89, 90, 92, 96
dialogue act 28–31, 33, 34, 89, 108
dialogue corpora 64, 95
dialogue corpus 23, 39, 40, 95
dialogue simulator 96
dialogue state 28, 30, 33, 34, 41, 48

E

environment 17, 18, 30, 31, 33, 34, 39, 41, 45–49, 51, 60, 69, 71, 75, 76, 78, 80–83, 86, 96, 108, 110, 111, 113–115
exploration 41
Exploration/Exploitation 41

F

feature 18, 21, 40, 42, 46, 49, 61, 86
features 49

G

greedy . . 41, 42, 58, 71, 73–76, 80, 89, 91, 92, 96, 113–115

H

handcrafted . 23, 27, 30, 60, 61, 63–66, 89, 90, 95, 96
handcrafted user 23, 50, 55, 59, 60, 63, 64, 66, 95
hang-up 96
hangup 23, 81, 89, 90, 96
hangup-model 96
human-model 64
human-model user . 23, 50, 55, 59–61, 63, 64, 66, 95, 96

J

jumpstart 22, 46, 48, 55, 65, 89

L

Lagrangian Relaxation 80
Lagrangian relaxation 86, 96
linear model 75
linear-regression 42

M

model 18, 19, 21, 22, 31, 39, 42, 46, 47, 49, 60, 61, 63–66, 75, 77–79, 81, 82, 86, 95, 110, 116
model-based 40, 116
model-free 40, 116

O

Off-policy 41
Offline 39, 42
On-policy 41
Online 31, 39, 41, 42, 71, 86, 95, 111, 113, 116
optimal 36–38

P

policy . . 17, 36–39, 41, 42, 48, 55–57, 61, 63–65, 78, 80, 96

R

regression 38–40, 75, 79
risk-neutral 76, 78, 82, 108
risk-sensitive 75, 78, 82, 108

S

safe policy 23, 50, 69, 87, 89, 96
semantic 18, 19, 22, 27, 29, 30
slot-filling . . 24, 28–30, 34, 35, 58, 80, 90, 108
speech recognition 19, 81
speech recognition error 81
stochastic 34, 57
stochastic policy 36
system act 30

T

task-oriented . . . 21, 22, 24, 30, 33, 34, 36
trajectory . . . 41, 57, 59, 60, 75, 76, 80–82
transfer . 18, 22, 23, 46–50, 55–58, 64–66, 87, 89, 92, 96, 113–116
Transition 57
transition 39–42, 46–51, 57, 58, 64, 66, 72, 75, 76, 78, 113–115
Turing 20
Turing-bombe 20
Turing-Machine 20
Turing-Test 20–22
turn 22, 36, 58, 70, 85, 108

U

user . 23, 27–31, 33, 34, 36, 39, 41, 45–51, 55–61, 63–66, 69, 80–82, 89, 90, 92, 95, 96, 108, 113, 114
user act 30
user adaptation 23, 24, 47–49, 55, 56, 59–61, 65, 66, 95
user-model 21, 49