Imitation hiérarchique et apprentissage par renforcement pour les systèmes de dialogue orientés tâches multi-domaines

Thibault Cordier

ACADÉMIE D'AIX-MARSEILLE
AVIGNON UNIVERSITÉ

THÈSE

Présentée à Avignon Université pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : INFORMATIQUE

École Doctorale 536 "Agrosciences & Sciences"
Laboratoire Informatique d'Avignon (EA 4128)

Hierarchical Imitation and Reinforcement Learning for Multi-Domain Task-Oriented Dialogue Systems

par
Thibault CORDIER

Soutenue publiquement le October 13, 2023 devant un jury composé de :

M.	Olivier PIETQUIN	Pr., Université de Lille / Google Research - CRIStAL	Rapporteur
M.	Stefan ULTES	Pr., Université Otto-Friedrich de Bamberg	Rapporteur
M.	Frederic BECHET	Pr., Université d'Aix Marseille - LIS	Examinateur
M.	Matthieu GEIST	Pr., Université de Lorraine / Google Research - LIEC	Examinateur
M.	Jean-Francois BONASTRE	Pr., Université d'Avignon - LIA	Examinateur
M.	Yezekael HAYEL	Pr., Université d'Avignon - LIA	Invité
M^{me}	Lina ROJAS-BARAHONA	Dr., Orange Labs	Co-Encadrante
M.	Tanguy URVOY	Dr., Orange Labs	Co-Encadrant
M.	Fabrice LEFEVRE	Pr., Université d'Avignon - LIA	Directeur de thèse

Remerciements

Le déroulement de cette thèse a représenté pour moi un projet de vie qui va au-delà de la réalisation académique. Comprendre l'essence de cette aventure nécessite de ne pas se limiter à observer le résultat final, mais aussi de considérer le chemin parcouru. Je suis profondément conscient que la réalisation de cette thèse n'aurait pas été possible sans le soutien inestimable de nombreuses personnes qui m'ont fait partager leurs connaissances et expertise et accordé leur confiance en mon travail.

Je tiens donc à remercier sincèrement tous ceux qui m'ont soutenu et suivi dans cette grande aventure, tout d'abord Fabrice Lefèvre pour la qualité de nos échanges, la pertinence de ses observations et son regard éclairé sur la recherche; Lina Rojas-Barahona pour son engagement et sa détermination sans faille et toute la passion qu'elle a su me transmettre; et Tanguy Urvoy, pour son enthousiasme et son désir de toujours approfondir les sujets complexes.

Je voudrais ensuite remercier Olivier Pietquin et Stefan Ultes d'avoir accepté d'être les rapporteurs de cette thèse ainsi que les examinateurs et invités de ma soutenance de thèse, Frederic Bechet, Matthieu Geist, Jean-Francois Bonastre et Yezekael Hayel.

Je tiens également à saluer tous ceux qui m'ont accompagné dans mon quotidien de doctorant, Rim Abrougui, Betty Fabre, Victor Guymard, Léo Jacqmin, Pierre Mahé, Mina Rafla et Colin Troisemaine et tout particulièrement Sébastien Montella sans qui mon expérience de la thèse n'aurait pas été la même.

Je tiens à ne pas oublier mes collègues d'Orange Labs, de l'équipe NADIA, en particulier Pierre Adam, Kévin Boeuf, Quentin Brabant, Eric Clavier, Gwénolé Lecorvé, Sonia Le Meitour, Constance Scherer, Morgan Veyret, Myriam Vialat, pour leur complicité et la dimension humaine qu'ils ont apportée et mes collègues du LIA, Antoine, Arthur, Gaëlle, Jarod, Lucas, Luis, Mathias, Paul-Gauthier, Salima, Thibault, Timothée pour leur accueil et leur sympathie.

Enfin, je souhaite terminer en remerciant particulièrement tous mes proches qui m'ont apporté un soutien indéfectible dans les moments de doute et de stress, ma compagne Xihui qui n'a pas eu peur de s'aventurer à mes côtés dans le périple d'une thèse en Bretagne et qui a été d'un soutien inestimable; mes frères Romain et Matthieu, qui ont été toujours présents quand j'avais besoin d'eux; et mes parents pour leur soutien inconditionnel qui m'a permis d'atteindre mes rêves.

Merci également à tous les autres que je n'ai pas pu citer mais qui auront laissé chacun à leur manière leur empreinte sur cette période de ma vie.

Résumé

Dans cette thèse de doctorat, nous étudions les systèmes de dialogue orientés tâches qui sont des systèmes conçus pour aider les utilisateurs à accomplir des tâches spécifiques, telles que la réservation d'un vol ou d'un restaurant. Ils s'appuient généralement sur un paradigme d'apprentissage par renforcement pour modéliser le dialogue permettant au système de raisonner sur les objectifs et les préférences de l'utilisateur, et de sélectionner les actions qui conduiront au résultat souhaité.

Malgré les avancées récentes, les systèmes de dialogue orientés tâches présentent encore plusieurs limites. L'une d'entre elles est la tendance de ces systèmes à échouer lorsque les utilisateurs s'écartent du comportement attendu ou introduisent de nouveaux objectifs au milieu de la conversation. Un autre problème est la difficulté de concevoir des systèmes robustes capables de gérer un large éventail de tâches.

Nous nous concentrons spécifiquement sur l'apprentissage à partir d'un nombre limité d'interactions, ce qui est crucial en raison de la rareté et du coût des interactions humaines. Les algorithmes standards d'apprentissage par renforcement nécessitent généralement une grande quantité de données d'interaction pour obtenir de bonnes performances. Pour relever ce défi, nous visons à rendre les systèmes de dialogue plus efficaces en termes d'échantillonnage dans leur entraînement.

Nous nous sommes inspirés principalement des idées d'imitation et de hiérarchie. Notre première contribution explore l'intégration de l'imitation dans l'apprentissage par renforcement. Nous nous appuyons sur la littérature existante qui souligne l'importance de l'imitation dans l'apprentissage, car les humains apprennent souvent en imitant des experts qui possèdent des connaissances précieuses. Nous étudions comment utiliser efficacement les démonstrations d'experts pour extrapoler les connaissances avec un effort de généralisation minimal. Alors que l'imitation s'avère efficace pour obtenir des performances et tirer parti de démonstrations réussies, nous observons des limites lorsqu'il s'agit de traiter une complexité plus élevée, en particulier dans le cadre d'un dialogue orienté tâches multi-domaines.

Notre deuxième contribution porte sur l'exploitation de la hiérarchie et de la structure inhérentes aux dialogues. En nous inspirant de l'avantage que présente la décomposition de problèmes complexes en problèmes plus simples, nous explorons la manière d'exploiter les similitudes entre les tâches et les domaines dans les systèmes de dialogue. En décomposant le problème principal en tâches élémentaires que nous maîtrisons, nous tirons parti de la hiérarchie pour résoudre efficacement des problèmes plus vastes et plus complexes. Cette approche permet d'économiser du temps de formation en partageant des stratégies entre des tâches similaires.

Enfin, nous consolidons nos résultats précédents et soulignons l'importance de l'apprentissage à partir d'un petit nombre d'interactions humaines dans les applications du monde réel. Les techniques d'apprentissage efficaces sur le plan de l'échantillonnage sont essentielles dans ce contexte, et nos recherches portent sur le développement de solutions efficaces dans le cadre de nos découvertes précédentes.

Abstract

In this Ph.D thesis, we study task-oriented dialogue systems that are systems designed to assist users in completing specific tasks, such as booking a flight or ordering food. They typically rely on reinforcement learning paradigm to model the dialogue that allows the system to reason about the user's goals and preferences, and to select actions that will lead to the desired outcome.

Despite these advances, there are still several limitations to task-oriented dialogue systems. One issue is the tendency of such systems to fail when users deviate from expected behavior or introduce new goals mid-conversation. Another issue is the difficulty of designing robust systems that can handle a wide range of tasks.

Our focus is specifically on learning from a limited number of interactions that is crucial due to the scarcity and costliness of human interactions. Standard reinforcement learning algorithms typically require a large amount of interaction data to achieve good performance. To address this challenge, we aim to make dialogue systems more sample-efficient in their training.

To guide our contribution journey, we draw from two main ideas: imitation and hierarchy. Our first contribution explores the integration of imitation with reinforcement learning. We build upon existing literature that emphasises the importance of imitation in learning, as humans often learn by imitating experts who possess valuable knowledge. We investigate how to effectively use expert demonstrations to extrapolate knowledge with minimal generalisation effort. While imitation proves efficient for achieving performance and leveraging successful trajectories, we observe limitations when dealing with higher complexity, particularly in multi-domain task-oriented dialogue.

Our second contribution focuses on harnessing the hierarchy and structure inherent in dialogues. Taking inspiration from the advantage of decomposing complex problems into simpler ones, we explore how to exploit task and domain similarities in dialogue systems. By decomposing the main problem into elementary tasks that we master, we leverage hierarchy to solve larger and more complex problems efficiently. This approach saves training time by sharing policies across similar tasks.

Lastly, we consolidate our previous findings and emphasise the importance of learning from a small number of human interactions in real-world applications. Sample-efficient learning techniques are essential in this context, and our investigation revolves around developing effective solutions within the framework of our previous discoveries.

Contents

Introduction

Since IBM's Deep Blue defeated world chess champion Garry Kasparov in a six-game rematch in 1997, Artificial Intelligence (AI) has continued to challenge human masters in historical events on the public scene. In 2011, IBM's Watson, a question-answering computer system capable of answering questions posed in natural language, competed on the quiz show Jeopardy! against champions Brad Rutter and Ken Jennings, winning the first-place prize. DeepMind's AlphaGo, a computer program that plays the board game Go, made its debut in 2015 and went on to defeat the European Go champion Fan Hui in a match. In March 2016, it made history by defeating Lee Sedol, one of the world's top Go players, in a five-game match. Cepheus, developed by the Computer Poker Research Group at the University of Alberta, became the first poker playing program that "essentially weakly solved" the game of heads-up limit Texas hold 'em. Additionally, AI has shown proficiency in mastering video games such as Atari, Dota 2 with OpenAI Five in 2017, and StarCraft II with AlphaStar in 2019. These achievements demonstrate the continuous advancement and potential of AI in various fields.

What is AI?

The term AI has become increasingly popular in recent years, but what does it really mean? Essentially, AI refers to the ability of machines to demonstrate intelligence, rather than relying solely on natural intelligence like animals and humans. The field of AI research involves the study of intelligent agents - systems that can perceive their environment and take actions that maximize their chances of achieving their goals. This pursuit can be viewed through two main lenses: one that focuses on thought processes and reasoning, and another that addresses behavior. Additionally, there are two approaches to measuring success - one that compares machine performance to human performance, and another that measures against an ideal concept of intelligence, known as rationality.

In practice, the focus on rational behavior in defining AI helps us develop models and algorithms to systematically solve problems. This approach is particularly useful in designing AI systems that tackle real-world problems with an objective of maximizing their chances of success. In simpler terms, AI can be defined as an automatic information processing tool or algorithm designed to achieve a specific goal.

Modern artificial intelligence techniques are ubiquitous AI is therefore applicable to any intellectual task and is widely used in our world today. Modern AI techniques are ubiquitous and too numerous to list here. In the 2010s, AI applications were at the heart of the most commercially successful areas of computing: search engines (such as Google Search), targeting online advertisements, recommendation systems (offered by Netflix, YouTube or Amazon), driving internet traffic, targeted advertising (AdSense, Facebook), virtual assistants (such as Siri, Google asisstant ot Alexa), autonomous vehicles (including drones and self-driving cars), turn-by-turn navigation, spell checker, word completion, automatic language translation (Microsoft Translator, Google Translate), facial recognition (Apple's Face ID or Microsoft's DeepFace), image processing, image labeling (used by Facebook, Apple's iPhoto and TikTok) and spam filtering.

Artificial intelligence techniques are extremely effective AI has become essential in various applications due to its many faculties, which make it efficient and effective. One of the advantages of AI is its **reliability**: it can perform tasks with consistent accuracy and do not tire or make mistakes due to fatigue or inattention. Another important facet of AI is **verification**: it can be programmed to follow specific rules and protocols, ensuring that they always operate within a defined set of parameters. AI can also provide continuous **monitoring** and analysis of vast amounts of data, which would be impossible for humans to perform manually. **Automation** is another key feature of AI, allowing repetitive or mundane tasks to be performed more efficiently and accurately by machines. **Decision support** is another crucial aspect of AI, where algorithms can analyze data and provide recommendations or insights to help humans make more informed decisions. AI can also provide **personalisation**, tailoring recommendations or services to individual users based on their preferences, behaviors, and past interactions. Finally, AI can provide **superhuman analysis**, allowing for the processing of vast amounts of data and performing complex computations beyond human capabilities.

Artificial intelligence benefits from advances in training models Machine Learning (ML) is a field of inquiry devoted to understanding and building methods that "learn" – that is, methods that leverage data to improve performance on some set of tasks. ML has rapidly evolved over the past few years, thanks to the emergence of Deep Learning (DL), methods based on artificial neural networks with

representation learning, and the advancements in computing capacities of modern machines, particularly Graphics Processing Unit (GPUs). This has led to significant improvements in the accuracy of deep neural networks, making them an essential tool for many fundamental tasks. Techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and transformers have revolutionized the fields of Computer Vision, Natural Language Processing (NLP), and Speech Recognition. The hardware advancements have allowed us to train these deep neural networks on vast amounts of data, leading to impressive results that were once thought to be impossible. With the continued development of machine learning techniques and hardware, we can expect further advancements in the field of AI in the years to come. Last year, OpenAI's language model called GPT-3 made a splash in our societies by setting a new standard for AI that can mimic human language and generate text that is almost indistinguishable from human writing with all the ethical issues that this implies.

AI for dialogue systems

Dialogue System (DS), also known as conversational agents or chatbots, are computer programs that engage in conversation with users. They can be used for a variety of tasks, including customer service, information retrieval, personal assistance, and entertainment. DS typically use NLP techniques to interpret user input and generate appropriate responses. In recent years, the field of DS has undergone rapid development, with advances in ML and NLP techniques leading to more sophisticated and effective systems.

A brief history of conversational AI The idea of machines that could converse with humans has a long history, dating back to science fiction stories such as Isaac Asimov's "I, Robot" and the character of Hal9000 in the film "2001: A Space Odyssey". However, the first significant milestone in the development of DS was the Turing Test, proposed by Alan Turing in 1950. The Turing Test is a measure of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. The test involves a human evaluator who engages in a natural language conversation with a machine and a human, without knowing which is which. If the evaluator cannot reliably distinguish between the machine and the human, the machine is said to have passed the test.

In the decades following the Turing Test, several rule-based DS were developed, which relied on handcrafted rules to generate responses. One early example was the ELIZA program developed in the 1960s, which used pattern matching and simple rules to simulate a psychotherapist. Another early DS was PARRY, developed in the 1970s, which simulated a paranoid schizophrenic patient. With the rise of ML in the 1990s, DS began to incorporate statistical methods to improve their performance. In particular, stochastic optimization algorithms were used to learn models of dialogue

behaviour from data. This led to the development of data-driven DS, which could learn from examples of human conversations to generate more natural and effective responses.

The current state of conversational AI Today, DS are used in a wide range of applications, including customer service, personal assistants, and social media. Some examples of DS include Apple's Siri, which can answer questions, set reminders, and perform various tasks on a user's behalf, Amazon's Alexa, which can control smart home devices, play music, and provide information on a wide range of topics, and Google Assistant, that can answer questions, set reminders, and control smart home devices, among other things. These systems use a combination of natural language processing (NLP), understanding (NLU) and generation (NLG) techniques and machine learning algorithms to interact with users and provide helpful responses.

DS are facing a number of challenges in their development, particularly in the areas of NLU, NLG and dialogue strategies. While recent advances in NLP techniques have made it more feasible to understand the nuances of human language, achieving direct control over dialogue strategy remains a difficult task. Large language models, such as GPT-3, have demonstrated remarkable ability in imitating human language, but designing effective dialogue strategies that can enable the system to engage in a meaningful conversation with the user and achieve its intended goals remains a challenge. To address this issue, researchers are exploring different approaches such as using ML algorithms to learn from previous interactions and optimise the dialogue strategy. However, this requires large amounts of high-quality training data, which can be a significant obstacle in certain domains.

Focus on task-oriented dialogue systems One area of particular interest in the field of DS is task-oriented dialogue systems. These are systems designed to assist users in completing specific tasks, such as booking a flight or ordering food. They typically rely on a Partial Observable Markov Decision Process (POMDP) to model the dialogue that allows the system to reason about the user's goals and preferences, and to select actions that will lead to the desired outcome.

One of the key challenges in developing task-oriented dialogue systems is the problem of Dialogue State Tracking (DST), which involves keeping track of the user's goals and preferences as the conversation progresses. This is a difficult problem because it requires the system to reason about the user's implicit intentions and beliefs, rather than relying solely on explicit dialogue cues. Another challenge is the design of effective Dialogue Policy (DP), which can balance the competing goals of providing helpful responses and achieving the system's objectives. In recent years, Reinforcement Learning (RL) has emerged as a promising approach to learn such policies, allowing DS to improve through interactions with users.

Despite these advances, there are still several limitations to task-oriented DS. One issue is the tendency of such systems to fail when users deviate from expected

behavior or introduce new goals mid-conversation. Another limitation is the difficulty of designing robust systems that can handle a wide range of tasks.

Despite these challenges, task-oriented dialogue systems continue to hold significant promise for a wide range of applications, including customer service, healthcare, education, and personal productivity. For example, task-oriented dialogue systems have been used to provide personalised recommendations for online shoppers and support language learning through conversation practice.

As research in conversational AI continues to advance, we can expect to see further progress in the development of more sophisticated and effective task-oriented dialogue systems, as well as the emergence of new applications and use cases. With the potential to revolutionise how we interact with technology and each other, conversational AI represents a crucial area of research and development for the future of computing and human society.

Important note

We precise that this thesis predates ChatGPT (released in November 2022) and therefore does not refer to these models of text-generation, now seen as conversational agents. This thesis is more about planning and task-oriented dialogue than dialogue fluidity. As impressive as it is, ChatGPT in its current version (no open-source) does not manage planning. We can expect to see task-oriented control plugins soon, but for the moment it's still a super chit-chat bot, which is prone to hallucinations and offers no solid guarantees in relation to trusted AI (the veracity of claims is an example).

Contributions

Our focus is specifically on task-oriented dialogue systems, where learning from a limited number of interactions is crucial due to the scarcity and costliness of human interactions. Standard RL algorithms typically require a large amount of interaction data to achieve good performance. To address this challenge, we aim to make dialogue systems more sample-efficient in their training.

To guide our contribution journey, we draw from two main ideas: imitation and hierarchy. Our first contribution explores the integration of imitation with RL. We build upon existing literature that emphasises the importance of imitation in learning, as humans often learn by imitating experts who possess valuable knowledge. We investigate how to effectively use expert demonstrations or expert supervision to extrapolate knowledge with minimal generalisation effort. While imitation proves efficient for achieving performance and leveraging successful trajectories, we observe limitations when dealing with higher complexity, particularly in multi-domain task-oriented dialogue.

Our second contribution focuses on harnessing the hierarchy and structure inherent in dialogues. Taking inspiration from the advantage of decomposing complex problems into simpler ones, we explore how to exploit task and domain similarities in dialogue systems. By decomposing the main problem into elementary tasks that we master, we leverage hierarchy to solve larger and more complex problems efficiently. This approach saves training time by sharing policies across similar tasks.

Lastly, we consolidate our previous findings and emphasise the importance of learning from a small number of human interactions in real-world applications. Sample-efficient learning techniques are essential in this context, and our investigation revolves around developing effective solutions within the framework of our previous discoveries.

In summary, this thesis aims to enhance task-oriented dialogue systems by addressing the challenge of sample efficiency. We achieve this through the integration of imitation and reinforcement learning, leveraging the benefits of hierarchy and structure in dialogue, and emphasising the ability to learn from a limited number of human interactions, which is crucial for practical applications.

Publications

The contributions discussed in the previous paragraph have led to the various publications in international and national conferences. Our first work (Cordier et al., 2020) has been shared through a remote oral presentation at the workshop of *Human in the Loop Dialogue Systems* of the Neural Information Processing Systems conference. The continuation of our work (Cordier et al., 2022b) has led to a poster presentation at the meeting of the *Special Interest Group on Discourse and Dialogue*. The same work (Cordier et al., 2022a) with a special focus have been shared via a poster presentation at the meeting of *Journées d'Études sur la Parole*. Our final work (Cordier et al., 2023) has been published in the findings of the *European Association for Computational Linguistics* via a presentation and a poster session.

Thibault Cordier et al. (2020). "Diluted Near-Optimal Expert Demonstrations for Guiding Dialogue Stochastic Policy Optimisation". In: *Proceedings of the Workshop of Human in the Loop Dialogue Systems, NeurIPS 2020*. URL: https://sites.google.com/view/hlds-2020/home

Thibault Cordier et al. (2022b). "Graph Neural Network Policies and Imitation Learning for Multi-Domain Task-Oriented Dialogues". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2022, Edinburgh, UK, 07-09 September 2022*. Ed. by Oliver Lemon et al. Association for Computational Linguistics, pp. 91–100. URL: https://aclanthology.org/2022.sigdial-1.10

Thibault Cordier et al. (2022a). "Et la robustesse ? ... bordel ! Comment les stratégies de dialogue par apprentissage structuré résistent aux bruits des entrées ?"

In: *Proceedings of Journées d'Études sur la Parole – JEP 2022*, pp. 501–510. URL: https://www.isca-speech.org/archive/jep_2022/cordier22_jep.html

Thibault Cordier et al. (2023). "Few-Shot Structured Policy Learning for Multi-Domain and Multi-Task Dialogues". In: *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*. Ed. by Andreas Vlachos et al. Association for Computational Linguistics, pp. 432–441. URL: https://aclanthology.org/2023.findings-eacl.32

Thesis Outline

This manuscript is organised as follows: In Part I, we provide an introduction of task-based dialogue systems and reinforcement learning required to understand the concepts used in this thesis. In Chapter 1, we propose a global overview on task-oriented dialogue systems and present the mathematical framework for its modeling. In Chapter 2, we discuss the main arguments for using reinforcement learning in a dialogue application and present the mathematical framework for its modeling. We conclude by giving an overview of more sophisticated mechanisms used for dialogue management and the challenges we address in the scope of this thesis.

In Part II, we list the contributions of the thesis to the scaling of dialogue systems for multi-domain task-oriented dialogues based on hierarchy and imitation techniques. In Chapter 3, we present several hybrid imitation and reinforcement learning strategies for single-domain dialogue policy. In Chapter 4, we present several structured policies based on hierarchical reinforcement learning and graph neural network that we combine with different degrees of imitation learning in order to effectively handle multi-domain dialogues. In Chapter 5, we focus on the ability to learn from a small number of human interactions that is crucial especially on multi-domain environments. We propose to use structured policies with full imitation to improve sample efficiency when learning on these kinds of environments.

Finally, the conclusion and perspectives in Part III puts an end to this thesis.

I

Task-Oriented Dialogue Systems

1

A Global Overview on Task-Oriented Dialogue Systems

In the introduction, we briefly addressed the evolution of conversational AI and its applications in our modern societies. We narrowed the scope of the study on task-oriented dialogue systems that are designed to achieve specific goals while conversing with humans, and possibly several goals during a unique dialogue.

In this chapter, we propose a global overview on task-oriented dialogue systems in order to provide key elements to understand the concepts used in this thesis. First, we briefly introduce some mechanisms inherent to human dialogues. Then we explain how to model a statistical spoken dialogue systems in particular by providing a probabilistic framework for dialogue simulation. To continue, we formalise the task-oriented dialogue problem and describe in detail the necessary concepts. Finally, we outline the approaches and tools needed for prototyping dialogue systems such as data collection and simulation.

For more details about all elements we introduce, we invite the interested reader to consult our main inspirational references on speech and language processing (Jurafsky and Martin, 2014), on task-oriented dialogue literature surveys (Paek et al., 2008a; H. Chen et al., 2017; J. Gao et al., 2018; Z. Zhang et al., 2020), on milestones in dialogue system modelling (J. Williams et al., 2007; Young et al., 2013b) and on pertinent applications (Rieser and Lemon, 2011).

1.1 From the perspective of a human conversation

Conversation between humans is a common activity that seems simple and natural at first glance, but it quickly becomes complex and complicated when examined more closely. Holding a conversation requires recognising and understanding speech and text, following the flow of the conversation, incorporating and extracting knowledge, deciding what action to take, structuring and generating the following speech. Before attempting to design a conversational system to dialogue with humans, it is crucial to understand the linguistic phenomena behind in this process. One good way to do

that is to study how humans converse with each others (Jurafsky and Martin, 2014). As an example, we are going to draw inspiration from a job interview between an employer and a candidate as illustrated in Figure 1.1. As if in **a game between two players**, a dialogue proceeds as follows: the employer takes a turn, then the candidate takes a turn, then the employer, and so on.

Turns A dialogue is a sequence of **turns** each hosting a single contribution from one speaker to the dialogue.

A turn can consist of a sentence, although it might be as short as a single word or as long as multiple sentences. By the way, a dialogue appears to be a **sequential decision making process**.

Initiative When a dialogue is controlled by one participant, we say that it has the conversation **initiative** (Walker and Whittaker, 1990).

As in a job interview, the employer asks questions and the applicant responds. However sometimes it is more common for initiative to pass from one participant to another. For instance, the applicant may ask a question to clarify something the employer said. In this case, we call such interactions **mixed initiative**. As we will see with human-machine interactions, we distinguish **user-initiative** and **system-initiative** dialogues and the situation will depend on the actions taken by each participant.

Dialogue acts Each utterance in a dialogue is a kind of action being performed by the speaker. These actions are commonly called **speech acts** or **dialogue acts** (Austin, 1975).

Due originally to the philosopher Wittgenstein (1953), the concept of *language-game* states that a word or even a sentence has meaning only as a result of the *rule of the game* being played. The philosopher Austin (1962) developed more fully the concept of *performative utterances* that are sentences which not only describe a given reality, but also change the social reality they are describing. In his framework, we distinguish:

- The **locutionary acts**, that are what was said and meant (*i.e.* the actual utterance and its ostensible meaning);
- The **illocutionary acts**, that are what was done (the semantic *illocutionary force* of the utterance, thus its real, intended meaning);
- The **perlocutionary acts**, that are what happened as a result (*i.e.* its actual effect such as persuading, convincing, scaring, inspiring);
- The **metalocutionary acts**, that categorise speech acts that refer to the forms and functions of the discourse itself (prosody and punctuation).

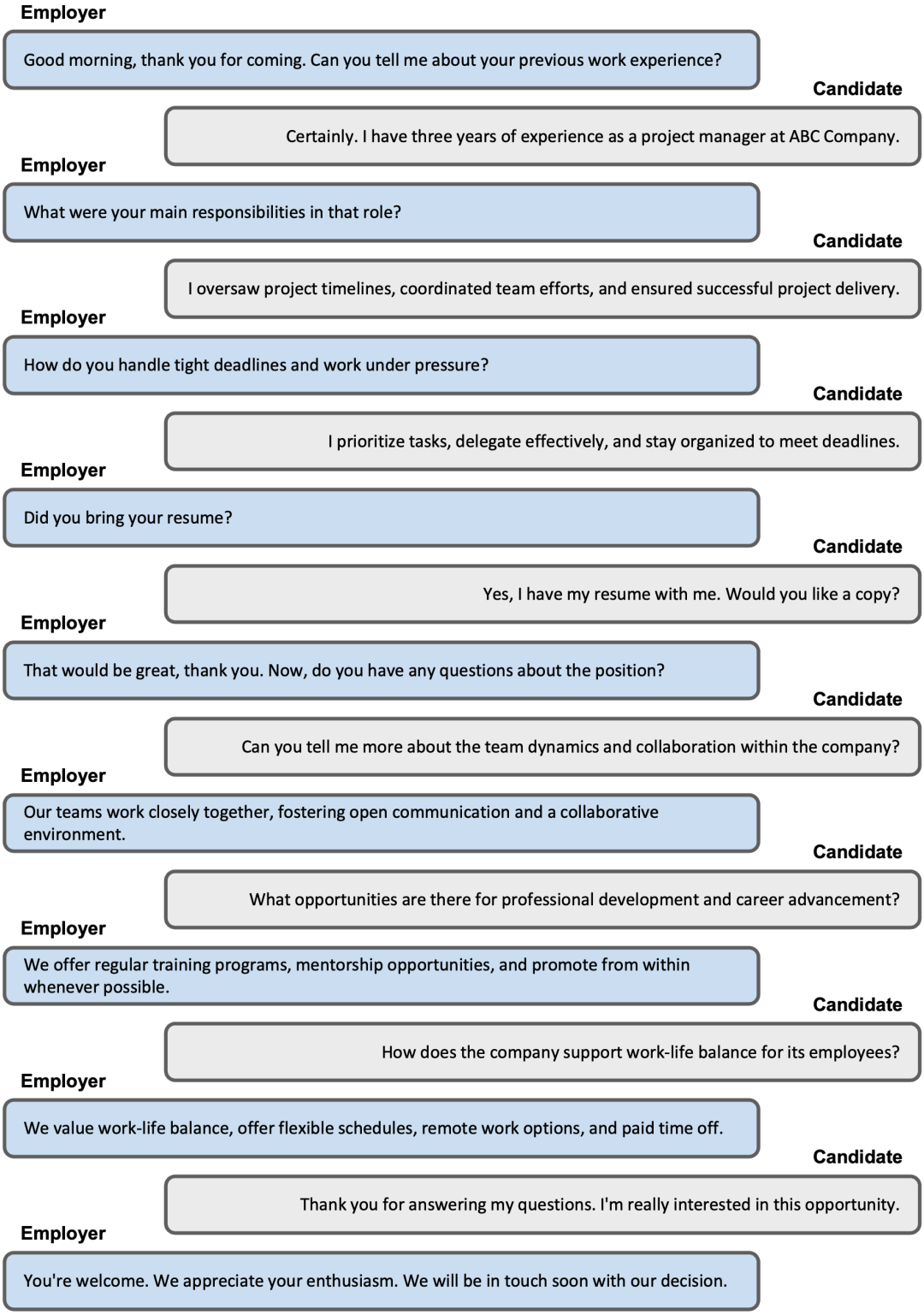


Figure 1.1: Example of a dialogue during a job interview between an employer and a candidate. The employer takes the initiative of the conversation and then the candidate in her turn. They communicate the information requested by the other in a structured way: the employee first questions the candidate and then vice versa.

When the employer says "did you bring your resume?' during the job interview, the illocutionary act is a request: "please give me your resume' even if the locutionary act (the literal sentence) was to ask about the presence of the resume. The perlocutionary act (the actual effect), could be to get the candidate to hand over his resume.

In what follows, we will focus on "illocution" when talking about "dialogue acts" since it is the driving force behind the conversation. Searle et al. (1969) set up the following classification of illocutionary speech acts:

- The **assertives** commit a speaker to the truth of the expressed proposition;
- The **directives** are to cause the listener to take a particular action (*e.g.* requests, commands and advice);
- The **commissives** commit a speaker to some future action (*e.g.* promises);
- The **expressives** (or acknowledgments) express on the speaker's attitudes and emotions towards the proposition with respect to some social action (*e.g.* congratulations, excuses and thanks);
- The **declarations** change the reality in accord with the proposition of the declaration (*e.g.* baptisms, pronouncing someone guilty or pronouncing someone husband and wife).

This classification is helpful to define what a dialogue system should understand and generate. Section 1.4 will introduce more formally the convention that we will use in our future experimentations.

Grounding It is necessary for participants to share a **common ground** to understand and be understood each other (Stalnaker, 1978).

A dialogue is not just a series of independent speech acts, but rather a collective act performed by the speaker and the listener. For instance, the applicant understands what the employer is asking and responds accordingly (and *vice versa*). This is because the participants share common concepts (referring to the language-game) and, in this way, implicitly determine what they agree on by *grounding* their statements one after another. Grounding means acknowledging that the listener has understood from the speaker; as when the employer confirms that the applicant has answered the question (Clark, 1996).

Moreover, and still in connection with the concept of the *language game*, the participants share and manipulate common concepts to communicate with each other. This may be communication rules accepted a priori or vocabulary known to all. Thus, when we design our dialogue systems, it will be important to define the context of the conversation (for whom and for what purpose).

Organisation and Structure Dialogue acts are organised in a logical sequence, bringing out **structures** in the conversation (Sacks et al., 1978).

Such local structures between speech acts are discussed in the field of conversational analysis (Sacks et al., 1978). The employer, after having asked a question (request act), expects an answer from the candidate (response act). A proposal is commonly followed by an acceptance or rejection. Dialogue acts appears to be present by pairs, called *adjacency pairs* (Schegloff, 1968).

These sequences usually appear successively, but may occur in parallel, revealing sub-parts of the dialogue called *side sequence* or *sub-dialogue* (Jefferson, 1972; Litman, 1985). The applicant may wish to correct a previous request by initiating a correction sub-dialogue. She may ask a clarification question which may initiate a sub-dialogue between a request and a response. Or she may ask to repeat a question that was not well understood because of a noisy environment.

Implicature Speakers attempt to be relevant in their interactions to guide their interlocutor and expect the other to draw **inferences** (Grice, 1975).

Inference helps the speakers in their understanding of the dialogue as pointed out by Grice (1975) in his theory of *conversational implicature*. Grice proposed that what enables hearers to draw these inferences is that conversation is guided by a set of maxims, general heuristics that play a guiding role in the interpretation of conversational utterances. For example, when the employer asks the applicant how long she has worked in the field, the applicant can list her experiences to show how much work she has done. She is indirectly answering the original question but hopes that her answer will guide the interviewer in the right direction.

Conclusion: These characteristics of human conversation (turns, speech acts, common ground, dialogue structure, initiative and implicature) show structural and dynamic aspects of a dialogue. As we will see, these elements will constrain our way of modeling a dialogue as a sequential turn-taking decision-making process between two conversational agents who share a common ground of knowledge and references. In particular, it will be necessary to define what these agents will talk about, what actions they will undertake and how they will translate the intentions of their interlocutor.

1.2 Modelling of statistical dialogue systems

Now that we have a better understanding of how human dialogues work, we propose to formalise and model a spoken dialogue system. This gives us a mathematical framework for further work and a procedure for how to design a dialogue system. First, we propose a functional description of the human-machine dialogue, and then we formalise it as a probabilistic framework.

1.2.1 Formal description of human-machine dialogue

For formalisation purposes, a human-machine dialogue can be considered as a sequential turn-taking process between a human user and a dialogue system controlled by the Dialogue Manager (DM) subsystem. It has access to an external information source, the Knowledge Base (KB) and can interact though speech and language processing subsystems such as Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Natural Language Generation (NLG), and Text-To-Speech (TTS) subsystems (Pietquin and Dutoit, 2006). We briefly introduce each subsystem of this dialogue process pipeline (J. Gao et al., 2018; Z. Zhang et al., 2020).

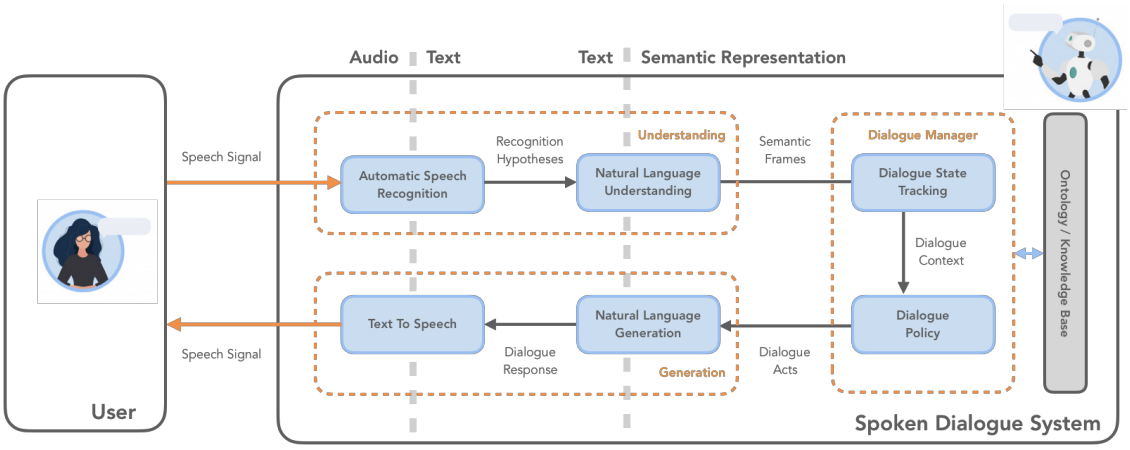


Figure 1.2: The complete pipeline of a spoken dialogue system consists of an understanding input process, a dialogue manager, and a generation output process.

Input processing subsystems Upstream of the decision making, the input modules process the signal, from speech to text and from text to a simplified semantic representation for the machine. These steps are performed in succession by ASR and NLU modules.

Automatic Speech Recognition (ASR) is an input module that enables the recognition and transcription of spoken language into text. It produces interpretation hypotheses as the most probable sentences the user could have said. Each hypothesis is associated with a confidence score that we call *ASR score*.

Natural Language Understanding (NLU) is an input module that extracts meanings from a text to build a semantic representation. It produces interpretation hypothesis as the most probable dialogue acts the user intended to perform, each one associated with a confidence score we call *NLU score*. More precisely, the NLU component takes the user utterance as input and maps the utterance to a structured semantic representation called *semantic frame*. This can be done by performing three tasks: *domain detection*, *intent determination*, and *slot tagging*. Typically, a pipeline approach is taken, so that the three tasks are solved one after another. We will detail these concepts more thoroughly in Section 1.3.

Decision-making subsystems The mastermind of the dialogue system is the **Dialogue Manager (DM)**. It is the control module that decides the best thing to say according to the current belief state of the dialogue. It is divided into two sub-modules: the DST and the DP.

Dialogue State Tracking (DST) maintains an internal dialogue state (also called *dialogue context*) by keeping track of the history of the dialogue. It usually takes the form of a confidence probabilities of each concept (*domain*, *intent* and *slot*) of the conversation.

Dialogue Policy (DP) decides what dialogue acts to take next according to its internal state and chooses some attributes (*slots* and *values*) to say to the user. It can access the KB in order to obtain the information needed to formalise its dialogue acts.

Output generation subsystems Downstream of the decision, the output modules transform the action into a human-understandable signal from dialogue acts to text and from text to speech. These steps are performed in succession by NLG and TTS.

Natural Language Generation (NLG) is an output module that transcribes the semantic representation (not only dialogue acts) of the current dialogue turn into a natural language text. The transcription process can be based on a predefined answer choice, on the completion of a template message or on a probabilistic word (or token) generation process.

Text-To-Speech (TTS) is an output module that enables the synthesis of text into speech. It consists of two parts: the front-end is responsible for assigning phonetic transcriptions to each word, and divides a text into sentences. The back-end, also called the synthesiser, is responsible for converting the symbolic linguistic representation into sound.

Extensive reading: Traditionally, dialogue systems are built in a modular architecture, so that each module is designed separately and combined together to achieve the overall goal. With the recent use of neural methods, which have the advantage of being differentiable and can be optimised by gradient-based methods such as back-propagation, fully data-driven end-to-end systems combine all modules (or a sub-set of these modules) into one that directly addresses the desired problem. The design may take longer and be more or less complex but benefits from less intermediation and therefore less propagation errors between modules. Interested readers can refer to J. Gao et al. (2018)'s survey to explore neural approaches to conversational AI.

1.2.2 Probabilistic description of a human-machine dialogue

We now propose a probabilistic framework of human-machine dialogues that will allow us to better understand how the modules communicate with each other and how external fluctuations influence the internal system during a dialogue turn. We will focus in particular on the probabilistic framework of the DM component.

How to formalise a human-machine dialogue turn? We detail how human-machine dialogue turns take place (J. Williams and Young, 2005; Pietquin and Dutoit, 2006). At each turn t, the DM generates a set of dialogue acts a_t that have to be transmitted to the user. In order to do that, the output generation subsystems (NLG and TTS) translate it successively into an intermediate text t_t and a synthesised spoken utterance sys_t. The human user answers by an utterance u_t built according to the dialogue acts she could extract from sys_t, to her knowledge k_t and to the goal g_t she tries to achieve. Then the input processing subsystems (ASR and NLU) process the user's utterance to produce an observation composed of the set of concepts o_t supposed to represent what the user meant and a set of scores m_t indicating the confidence of those systems in their processing coming from both the ASR (CL_{ASR}) and the NLU (CL_{NLU}) modules. The observation can be considered as the result of the processing of the dialogue acts by its environment. The DM finally uses o_t to update its internal state s_{t+1} thanks to a model of the task. During the input and output processing, both the user's and the system utterances are mixed with external noise n_t. There are multiple causes of ambient noise such as user's indecisions generating agramaticality, repetitions, etc. or incorrect automatic processes in speech generation and understanding. An illustration of this pipeline for a turn is presented in Figure 1.3.

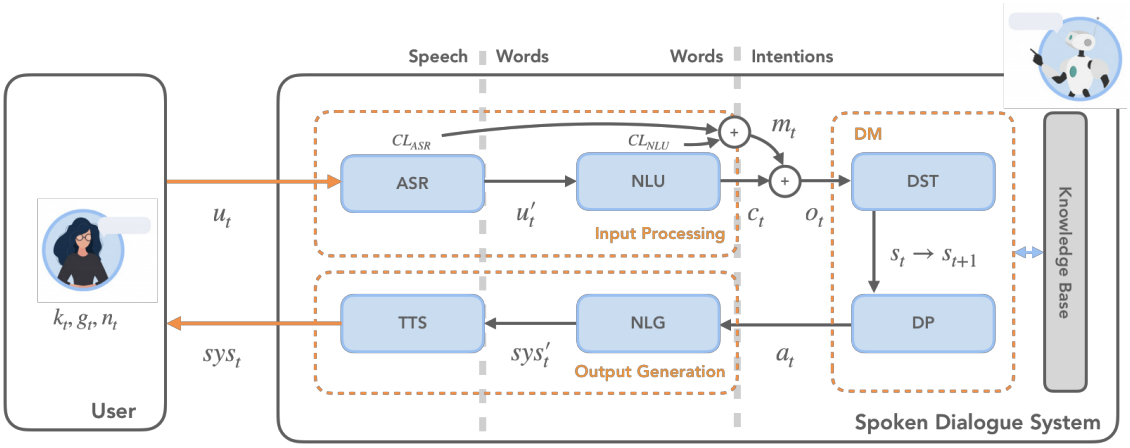


Figure 1.3: For a dialogue turn, the spoken dialogue system will transform step by step the input (audio or textual) into a semantic representation to then make a decision and perform the reverse transformation. Inspired from Pietquin and Dutoit (2006).

Notation	Definition
t	the time step or turn number
s_t	the internal system state at time t
a_t	the set of system dialogue acts at time t
sys_t	the textual (or spoken) system utterance at time t
u_t	the textual (or spoken) user utterance at time t
k_t	the user's knowledge at time t
g_t	the user's goal at time t
o_t	the system observation (or the set of user dialogue acts) at time t
CL_{ASR}	the confidence level of ASR module at time t
CL_{NLU}	the confidence level of NLU module at time t
m_t	the set of confidence scores on o_t at time t
n_t	the external noise at time t

Table 1.1: Notations used and corresponding definitions for a probabilistic description of a human-human dialogue.

A probabilistic framework for SDS: Based on these elements, the functioning of the dialogue system can be described by the joint probability of all signals occurring during an interaction given the current state and the noise. In particular, it is possible to factor this probability by taking into account their time dependence. According to the used notations we recall in the Table 1.1, this allow us to propose the following probabilistic framework of SDS:

Definition 1.1: Probabilistic framework for SDS

The functioning of the system can be represented by the following factored joint probability:

$$\begin{aligned} P(s_{t+1}, o_t, a_t, u_t, sys_t, g_t, k_t, |s_t, n_t) = \\ \underbrace{P(a_t|s_t)}_{\text{DM}} \cdot \underbrace{P(sys_t|a_t, s_t, n_t)}_{\text{NLG + TTS}} \\ \cdot \underbrace{P(k_t|sys_t, s_t, n_t)}_{\text{Knowledge Update}} \cdot \underbrace{P(g_t|k_t)}_{\text{Goal Modification}} \\ \cdot \underbrace{P(u_t|g_t, k_t, sys_t)}_{\text{User Utterance}} \\ \cdot \underbrace{P(o_t|u_t, a_t, s_t, n_t)}_{\text{ASR + NLU}} \cdot \underbrace{P(s_{t+1}|o_t, s_t)}_{\text{DST}} \end{aligned} \tag{1.1}$$

We use Definition 1.1 to extend our probabilistic framework to DM. We propose to simplify the joint probability by marginalising the DM components. We state that noise only has impacts in the environment and that the internal state representation depends only on the last observation of the environment.

Definition 1.2: Probabilistic framework for DM

From the point of view of the DM, one can describe the system functioning by a joint probability. By omitting the time indexes and denoting s' the state at time $t+1$, one can write:

$$P(s', o, a|s, n) = \underbrace{P(a|s)}_{\text{DM}} \cdot \underbrace{P(o|a, s, n)}_{\text{Simulation Env.}} \cdot \underbrace{P(s'|o, s)}_{\text{DST}} \tag{1.2}$$

If we isolate the DM module, we see that the observation can be considered as the result of the processing of the DM dialogue acts by its environment.

1.3 Task-oriented dialogue problem formulation

For now on, we specifically address task-oriented dialogue problems. We formalise this type of problem and present the key elements to understand the field that we will use as a common thread for the thesis.

1.3.1 Focus on information-seeking dialogue systems

Above all, task-oriented dialogues are centered around a specific subject with a clearly defined objective unlike chitchat scenarios. Here, we propose to focus on the emblematic use case of **information-seeking** task. We are going to show that information-seeking dialogue strategies can serve as an interactive natural language interface to database search (Androutsopoulos and Ritchie, 2000).

Let us keep the dialogue strategies aside and take the example of a service that provides access to a database containing a large number of search candidates. Here, the concept of **search candidates** is, in a broad sense, a list of candidates that response to a request of the user. The challenge is therefore to allow users to find different candidates satisfying their constraints, to choose a result and to request the needed information. Many services offer this functionality from simple slot-filling tasks, such as product ordering or service reservation (J. Williams et al., 2007), to more complex assistance tasks, such as tutoring tasks (Tetreault et al., 2006), technical support or customer service that advise and assist in case of problems. The simpler and usual approach consists of proposing a form containing boxes to be filled in with the desired constraints. The advantage of this approach is that it is simple to understand and not prone to errors. However, the downside is that the search procedure may not be optimal and can be time-consuming for the user.

Information-seeking is a task that is already done automatically by machines, nevertheless it can be done more efficiently by an AI that drives the questions on the form to browse the database more efficiently. These interactions between machines and humans can be done through natural language which can help to get away from the cumbersome nature of forms. From this perspective, we see that the dialogue is

used as a proxy for the search form and thus makes it possible to automate the task, makes the search for information more natural and also allows a spoken interaction.

This use case allowed us to better understand the characteristics of a task-oriented dialogue. Therefore we propose the following definition:

Definition 1.3: Task-oriented dialogue

We call a conversation involving a user and an assistant for *information-seeking* a **task-oriented dialogue**:

- The assistant provides a service in which she is responsible for solving a predefined information-seeking task.
- The user is looking for a solution that can answer her problem defined by a structured set of constraints.

Such conversation gives the opportunity to the interlocutors to share their information to find a solution to the search. To be more specific, information-seeking is the process of attempting to obtain information that can be decomposed in two major phases (Rieser and Lemon, 2011):

- The **information acquisition** is the initial phase during which the system is gathering information from the user by requesting her constraints;
- The **information presentation** is the second phase that starts when the system has obtained sufficient information that matches the user's requirements. The system is now ready to present the candidates to the user, detailing the information needed to solve the problem.

1.3.2 Elements of task-oriented dialogue problems

We propose to setup the main elements relative to a task-oriented dialogue.

Tasks and Domains Task-oriented dialogue systems search to satisfy users' needs by accomplishing a particular task for a specific domain.

Usually, a **task** refers to the goal of the system - *e.g.* informing, booking, selling - and it is related to a specific **domain** that refers to the set of concepts and values speakers can talk about - *e.g.* restaurants, attractions, hotels, train, laptops. For example, in a dialogue about booking train tickets, we can talk about the notions of departure and arrival cities which can take as values the cities served by the transport.

A dialogue is said to be **single-domain** if the task to be completed concerns only one domain. It is said to be **multi-domain** (respectively **multi-task**) if it can be broken down (sequentially or in parallel) into sub-dialogues related to distinct domains (respectively distinct tasks).

Slots A task-oriented dialogue is limited by the set of concepts speakers can talk about, called **slots**.

We refer to a widely used taxonomy, originally formalised in the Dialogue State Tracking Challenge (DSTC), to describe the information-seeking problem (J. Williams et al., 2016). By standing on the side of the user:

- a slot is said **informable** if the user can provide a value for, to use as a constraint on their search (for the system, this kind of slot is requestable).
- a slot is said **requestable** if the user can ask the system the value of a slot (for the system, this kind of slot is informable).

On principle, all informable slots are also requestable. In the example of the train ticket booking domain, the departure city is an informable slot 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 (Carrara, 2019).

Frames Task-oriented dialogue system must manage an internal state to follow the evolution of the dialogue. Such internal states are called **frames**.

A **frame** is a kind of knowledge structure representing the intentions the system can extract from user sentences. It consists of a collection of slots, each of which can take a set of possible values. Its role is to keep in memory the knowledge acquired by the system and the dialogue history as a *picture* at a specific time in the conversation. Each domain defines its own structure of frames, sometimes called a **domain ontology** (Jurafsky and Martin, 2014).

User and system acts **Dialogue acts**, as the abstract level of how utterances change the conversational state, are a good level of description for any communication between agents. (Traum, 1999).

Dialogue acts are formalised as predicates, possibly with slots or slot-values pairs as arguments. Some acts are common to every dialogues such as HELLO, REPEAT or BYE. Others acts depend on the domain ontology such as INFORM or REQUEST as they are direct instances of the requestable and informable slots. Examples of dialogue acts specific to a restaurant search task are REQUEST(slot=food) or INFORM(slot=price, value=moderate).

Here, we describe the most used dialogue act in dialogue system frameworks:

- REQUEST(slot) is used to request the value of a requestable slot;
- INFORM(slot, value) is used to inform the value of an informable slot;
- SELECT(slot, set of values) is used to inform a set of possible values for a slot and ask to select one of them;

- RECOMMEND(set of slot-value pairs) is used to make a recommendation by specifying a set of slot-value pairs;
- WELCOME / GREET / BYE / REPEAT / REQMORE are common dialogue acts for acknowledgement or misunderstanding.

This is not an exhaustive list of dialogue acts and the conventions used may differ from one application to another. However, they are common to the majority of dialogue systems, as we shall see later.

User goal Task-oriented dialogues can be reduced to a slot-filling problem and a value-providing problem. The user's **goal** is determined by her constraints and expectations.

Indeed, the constraints and expectations of the user are unknown to the system but they can be formalised in the dialogue frame (which can be seen as a form), that the system will fill for the user by questioning him and by querying the database. Then, the objective of the system is to match its dialogue frame with the user's hidden goal. From this point of view, the user goal can be defined as a set of constraints by slot-value pairs, *e.g.* {food = chinese, area = east}, and as a set of expectations, *e.g.* {address = ?, phone = ?}.

Similarly, we define the **gold goal** which corresponds to a goal with explicit expectations. In fact, it is possible that the system finds several search candidates that respect the constraints provided by the user. It is then assumed that the user expects to find a unique offer which is unknown to him but which he will recognise eventually and he will stop as consequence the search process.

1.4 Evaluation, dialogue collection and simulation

After having detailed the mechanism underlying the dialogue and the characteristics of a task-oriented dialogues, we now propose to work on the means of evaluating, collecting and simulating dialogues for development and testing.

1.4.1 Evaluation metrics for task-oriented dialogue

Dialogue evaluation is an indispensable stage for measuring the capacity of a system to solve a task. Because a task-oriented dialogue system is user-centered, it could be interesting to get an idea of the user satisfaction by computing a *user satisfaction rating*. A possible protocol could be having users interact with a dialogue system to perform a task and then having them complete a questionnaire. However, in practice, it is infeasible to run complete user satisfaction studies because of economical and technical issues. It seems unreasonable to evaluate a system like this systematically after every change. That is why it is useful to have performance evaluation heuristics that correlate well with human satisfaction (Jurafsky and Martin, 2014).

The most common choice to evaluate dialogue systems is to use measures based on *task success* (or *task completion*). We previously state that task-oriented dialogues are related to information-seeking task. We can then evaluate the system by analysing whether it properly provided the information the user has requested.

Task success measures To assess how well a task is solved, we can use precision, recall and F-score measures: **recall** evaluates whether all the requested information has been informed; **precision** evaluates whether only the requested information has been informed; **F-score** is the harmonic mean of recall and precision.

For a given dialogue, by denoting the *true positive* TP as the relevant retrieved information, the *false positive* FP as the non-relevant retrieved information and the *false negative* FN as the relevant non-retrieved information, we define the recall, the precision and the F-score as follows:

$$\text{recall} = \frac{|\{\text{relevant information}\}| \cap |\{\text{retrieved information}\}|}{|\{\text{relevant information}\}|} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{|\{\text{relevant information}\}| \cap |\{\text{retrieved information}\}|}{|\{\text{retrieved information}\}|} = \frac{TP}{TP + FP}$$

$$\text{Fscore} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

We define the **average recall**, the **average accuracy** and the **average F-score** as the average of these measures over the dialogues.

Thus, from a global point of view, we can propose to compute a *task success rating* which estimates how successful a dialogue system performs the task. A dialogue is marked as **successful** if all the requested information has been informed *i.e.* the proposed entity meets all the constraints specified in the *gold user's goal*. It is possible to relax this criterion by considering a dialogue as **completed** if it is successful from the user's point of view *i.e.* the proposed entity meets all the constraints specified in the *standard user's goal*.

Task success rating The most common choice to evaluate dialogue system is the **task success rate** (respectively **task complete rate**) that is the fraction of successful (respectively completed) dialogues that solves the user's problem.

At the same time, we prefer when the task is solved in a few turns. This can be evaluated by an efficiency cost measured by the total elapsed time *e.g.* the total number of elapsed dialogue turns.

1.4.2 An overview on task-oriented dialogue collection

We now explain how to generate and collect dialogues and for what purpose it can be useful. A dialogue dataset should approximate real human-human interactions in order to facilitate development and testing. We distinguish three groups of dialogue collection approaches:

- **Machine-to-Machine** approach consists on creating an environment where the user and system roles are simulated to generate a complete conversation flow with predefined dialogue templates (Bordes et al., 2016; Shah et al., 2018; Rastogi et al., 2020);
- **Human-to-Machine** approach consists on generating dialogues based on an existing dialogue system in front of a human user (Raux et al., 2005; J. Williams et al., 2013);
- **Human-to-Human** approach consists on directly collecting a large amount of human-human conversations (Ritter et al., 2010; Schrading et al., 2015; Lowe et al., 2015).

In the field of Human-to-Human interactions, a common technique for collecting more realistic dialogues is to simulate human users interacting directly with a dialogue system, called the Wizard-of-Oz (WOZ) setup (Kelley, 1984; Fraser et al., 1991; Dahlbäck et al., 1993). A large number of dialogue collection are based on this setup (T.-H. Wen et al., 2016; Asri et al., 2017; Pawel Budzianowski et al., 2018). To be more precise:

Wizard-of-Oz experiment In a **Wizard-of-Oz** experiment, a human operator, the so-called *wizard*, simulates the behaviour of a dialogue system, while the user is lead to believe that she is operating with a real system (Rieser and Lemon, 2011).

WOZ studies are performed to collect data before developing a dialogue system. These data corpus were collected using crowd-sourcing in which a different worker was computing one turn at the time. As we will see later, one of the advantages of this technique is that they allow us to 'bootstrap' a dialogue system in ideal conversations through expert demonstrations. They are opposed to real human-computer interactions, as the wizard has perfect recognition and understanding of the user's utterance, unlike a prototype dialogue system. Likewise, it is possible to study the behaviour of the human operator and give an insight into the preferences of the human user.

In the following, for future experiments, we focus on one specific dataset namely Multi-Domain Wizard-of-Oz (**MultiWOZ**) dataset (Pawel Budzianowski et al., 2018) that was collected with a WOZ experiment. Since its release, it has been improved and corrected by Eric et al. (2020), Zang et al. (2020), Han et al. (2021), and Ye et al. (2021).

MultiWOZ The **Multi-Domain Wizard-of-Oz** dataset is a large annotated and open-sourced collection of simulated human-human conversation between a tourist and a clerk from an information center in a touristic city. It includes different scenarios based on two basic tasks - *Requesting* and *Booking* (referred to as *find* and *book* tasks) - and it includes seven possible domain ontologies - *Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train.*

This corpus is based on a scenario sampling procedure from predefined domain and task templates. In particular, some scenarios propose to change the goal during conversations by simulating the mismatch of initial constraints with the database, which leads the user to follow his goal with alternative constraints. This results in more realistic conversations with single or multi-domain and task scenarios that are naturally connected to each other. For example, a tourist needs to find a hotel, to get the list of attractions and to book a taxi to travel between both places. Table 1.2 presents the global ontology with the list of considered dialogue acts as presented in the paper of Pawel Budzianowski et al. (2018).

Domains & Tasks	Informable slots / Requestable slots
Restaurants ↳ find, book	area, pricerange, food, postcode, bookday, bookpeople, booktime / name, address, phone, ref
Attraction ↳ find	area, type, entrancefee / name, address, postcode, phone, openhours
Hotel ↳ find, book	area, type, pricerange, parking, internet, stars, bookpeople, bookday, bookstay / name, address, phone, ref
Taxi ↳ book	destination, departure, arriveby, leaveat / phone, type
Train ↳ find, book	destination, departure, day, arriveby, leaveat, bookpeople / trainid, ref, price, duration
Hospital ↳ find	department / address, phone, postcode
Police ↳ find	- / name, address, phone, postcode
Dialogue acts	inform, request, select, recommend, not found, request booking info, offer booking, inform booked, decline booking, welcome, greet, bye, reqmore

Table 1.2: Description of MultiWOZ dataset by presenting its different domain ontologies and the considered dialogue acts

1.4.3 Focus on task-oriented dialogue simulation

We have previously presented different approaches for collecting "static" dialogues. It is still possible to generate "dynamic" human-machine interactions with prototype dialogue systems. Although this approach is necessary to provide a relevant solution for human users, it is economically and technically expensive to be implemented in practice. A natural way to get around this challenge is to build a **simulated user**, with which a dialogue system can directly interact at virtually no cost. We present in the following dialogue environments for prototype systems and summarise their main characteristics.

Dialogue simulation is used for different purposes, for example for testing and debugging prototypes systems or for automatic strategy development. Therefore, the use of simulated dialogue environments has helped to overcome the problem of data collection and to design dialogue systems more quickly.

In order to build a simulated user, it is possible to adapt the same dialogue pipeline presented in Section 1.2 to model the user's behaviour. The input processing and output generation components of the simulator can be implemented in the same way. The degree to which the simulated interaction resembles real human-computer interaction depends on the quality of these components. However, as we are specifically interested in the design of the DM module, we will first abstract from these external components and reduce the interaction to the level of intent where the system and the simulated user directly exchange dialogue acts. An error model will be used to simulate error-prone pipeline components. We therefore highlight two approaches to model the user's DM (Schatzmann et al., 2006):

- **Model-based simulation** is an approach based on building user simulators entirely on data (Eckert et al., 1997; Levin et al., 2000; Chandramohan et al., 2011b; Asri et al., 2016).
- **Agenda-based simulation** is a popular approach based on handcrafting hidden agenda-based user simulators (Schatzmann et al., 2009) as instantiated in Xiujun Li et al. (2016), Ultes et al. (2017a), and Zhu et al. (2020).

One obvious limitation is that it raises the chicken-and-egg problem, *i.e.* the problem of first having to build a user simulator to build an assistant simulator and vice-versa. In the same time, the evaluation of the user simulator is not obvious (Pietquin and Hastie, 2013), and it is often observed that dialogue systems overfit to particular user simulator. The gap between user simulators and humans is thus the main limitation of these approaches. However, agenda-based user simulation can be built without access to a dialogue dataset and has the advantage of having an explainable behaviours. In practice, it has been shown that they produce sufficiently realistic dialogue behaviours. All these aspects are relevant in the development and testing of prototype systems. For the reasons stated above, we use the agenda-based simulation in the scope of this thesis.

Agenda-based simulation An agenda-based approach is a probabilistic method for simulating user behaviour based on a compact representation of the **user goal** and a stack-like **user agenda**:

- Each dialogue simulation starts with a randomly generated user goal that is unknown to the dialogue system;
- At each turn, the user simulator acts on a stack-like user agenda which is a structure containing the pending user dialogue acts that are needed to elicit the information specified in the user goal.

In this work, we used two popular simulated dialogue frameworks with agenda-based user and hand-crafted agent simulators: PyDial and ConvLab.

PyDial for single-domain single-task-oriented dialogues: Proposed by Ultes et al. (2017b), PyDial is an open-source statistical spoken dialogue system toolkit implemented in Python[1], which provides domain-independent implementations of statistical approaches for all dialogue modules, as well as simulated users and simulated error models. It is pre-loaded with a total of ten domains of differing complexity.

In practice, the framework is used for single-domain simulation as proposed in the benchmarks of Casanueva et al. (2017), that focuses on three domains, from the simplest to the most complex: *Cambridge Restaurants*, *San Francisco Restaurants* and *Laptops*. Furthermore, to test the capability of algorithms in different environments, a set of tasks has been defined that spans a wide range of environments across a number of dimensions. In total, six environments with different user model/error model/action mask settings are defined, representing environments with different SER, with possibly an action masking (*i.e.* heuristics which reduce the number of actions the agent can take in each dialogue state) and with different user behaviour model, from standard to unfriendly configurations (*i.e.* where the users barely provide any extra information to the system). We summarise these environment configurations in Table 1.3. Table 1.4 provides a summary of the characteristics of each domain and Table 1.5 a summary of the characteristics of a dialogue state[2].

Env	Env.1	Env.2	Env.3	Env.4	Env.5	Env.6
SER	0%	0%	15%	15%	15%	30%
Masks	On	Off	On	Off	On	On
User	Standard	Standard	Standard	Standard	Unfriendly	Standard

Table 1.3: Set of benchmarking tasks. Each user model/error model/action mask environment is evaluated in the three different domains.

[1]Deprecated link we used for PyDial in Python2: http://pydial.org/. New link to access PyDial in Python3: https://pydial.cs.hhu.de/.

[2]In the following, we refer to a dialogue frame as a dialogue state.

Domain	# of informable slots	# of requestable slots
PyDial framework	find task	find task
Cambridge Restaurants	3	9
San Francisco Restaurants	6	11
Laptops	11	21
ConvLab framework	find/book task	find task
Restaurants	4/3	5
Attraction	3/-	7
Hotel	7/3	5
Taxi	4/-	2
Train	5/1	5
Hospital	1/-	3
Police	-/-	3

Table 1.4: Domains Description of PyDial and ConvLab framework

ConvLab for multi-domain multi-task-oriented dialogues: Initially proposed by Lee et al. (2019) and followed by an improved version of Zhu et al. (2020), the ConvLab framework is presented as an open-source toolkit in Python[3] that enables researchers to build multi-domain multi-task-oriented dialogue systems and perform end-to-end evaluations and system-wise evaluations. Notably, they have been proposed to support several large-scale dialogue datasets including an extended version of the dataset MultiWOZ (Eric et al., 2020) with user dialogue act annotations.

In practice, the framework features the MultiWOZ task based on a scenario sampling procedure from predefined domain and task templates similar to that used for MultiWOZ data collection. Indeed, the sampling frequency and the correlation of occurrence between domains and tasks are preserved, as well as some scenarios propose to change the objective during the conversations by simulating the mismatch of the initial constraints with the database. Furthermore, it has been used as the standard platform for the multi-domain task-completion dialogue track in DSTC8[4] (Kim et al., 2021) and as a benchmarking tool to provide a global system-wise evaluation (Takanobu et al., 2020). Online benchmarks are available for the system-wise and the end-to-end evaluations with ConvLab2 tool in the corresponding Github repository, as well as with the MultiWOZ evaluator in the corresponding Github repository. Table 1.4 provides a summary of the characteristics of each domain of MultiWOZ and Table 1.6 a summary of the characteristics of a dialogue frame aligned with the dialogue features of MultiWOZ.

[3]Link we used for ConvLab2 in Python3: https://github.com/thu-coai/ConvLab-2

[4]Link to access ConvLab used for DSTC8 Track 1 Task 1 "End-to-End Multi-Domain Dialog Challenge": https://github.com/ConvLab/ConvLab

Component & Description
Beliefs
constraint slot beliefs: $\{b_s^{inf} \in \Delta\mathcal{V}_s, \forall s \in \mathcal{S}^{inf}\}$ The goal constraint belief for each informable slot. This is either an assignment (probability) of a value from the ontology which the user has specified as a constraint, or is a special value — either *dontcare* which means the user has no preference, or *none* which means the user is yet to specify a valid goal for this slot.
request slot beliefs: $\{b_s^{req} \in \Delta\mathbb{B}, \forall s \in \mathcal{S}^{req}\}$ A set of requested slots, *i.e.* those slots whose values have been requested by the user, and should be informed by the system.
discourse act belief: $b^{act} \in \Delta\mathcal{D}^{act}$ An assignment of the current dialogue act of the user. This is one of - *hello* - *repeat* - *silence* - *acknowledgement* - *thank you* - *bye* - or *none* if the user is requesting or informing a constraint.
method belief: $b^{meth} \in \Delta\mathcal{D}^{meth}$ An assignment of the current dialogue search method. This is one of – *by constraints*, if the user is attempting to issue a constraint, – *by alternatives*, if the user is requesting alternative suitable venues, – *by name*, if the user is attempting to ask about a specific venue by its name, – *restart*, if the user wants to restart the call – *finished*, if the user wants to end the call – or *none* otherwise.
Features
last informed venue: $f_1 \in \mathcal{V}_{DB(d)}$ The name of the last venue offered by the system to the user w.r.t the constraint slots.
informed venue since none: $f_2 \in list(\mathcal{V}_{DB(d)})$ The name list of the previous venues offered by the system to the user w.r.t the previous constraint slots.
last action inform none: $f_3 \in \mathbb{B}$ A boolean showing that during the last turn, the system couldn't respond to the user's request.
offer happened: $f_4 \in \mathbb{B}$ A boolean showing that during the last turn, the system made a offer to the user.
inform info: $f_5 \in \mathbb{B}^5$ The vector counting the number of entities *count* matching with *numAccepted* slots in acceptance list: [count==0, count==1, 2<=count<=4, count>4, discriminatable]. Here, discriminatable is a boolean showing that there is a question which we could ask which would give differences between the values.
User Acts
user acts: $a^{user} \in list(\mathcal{A}^{user})$ The list of the last user actions with potentially corresponding probability.
Dialogue History
dialogue history: $h \in list(list(\mathcal{A}^{sys}) \times list(\mathcal{A}^{user}))$ The list of the previous couple of system and user composite actions.

Table 1.5: Belief State Template of PyDial framework

Component & Description
Beliefs
constraint slot beliefs: $\{b^{inf}_{d,s} \in \mathcal{V}_s, \forall s \in \mathcal{S}^{inf}_d, \forall d \in \mathcal{D}\}$ The goal constraint belief for each informable slot. This is either an assignment of a value from the ontology which the user has specified as a constraint, or is a special value — either *dontcare* which means the user has no preference, or *none* which means the user is yet to specify a valid goal for this slot. To be exact, for each domain, the constraint slot dictionary separates slots w.r.t the task *i.e.* we distinguish the *find* slot dictionary and the *book* slot dictionary. request slot beliefs: $\{b^{req}_{d,s} \in \mathbb{B}, \forall s \in \mathcal{S}^{req}_d, \forall d \in \mathcal{D}\}$ A set of requested slots, *i.e.* those slots whose values have been requested by the user, and should be informed by the system.
Features
terminated: $f_1 \in \mathbb{B}$ A boolean showing that the user wants to end the call. booked: $f_2 \in \mathcal{V}_{DB(d)}$ The name of the last venue offered by the system to the user w.r.t the constraint slots with additional information like reference. To be exact, this feature is located in the *book* slot dictionary. degree pointer: $f_3 \in \mathbb{B}^6$ The vector counting the number of entities *count* matching with constraint slots in acceptance list: [count==0, count==1, count==2, count==3, count==4, count>=5].
System Acts
system acts $a^{sys} \in list(\mathcal{A}^{sys})$ The list of the last system actions.
User Acts
user acts $a^{user} \in list(\mathcal{A}^{user})$ The list of the last user actions.
Dialogue History
dialogue history $h \in list(list(\mathcal{A}^{sys}) \times list(\mathcal{A}^{user}))$ The list of the previous couple of system and user composite actions.

Table 1.6: Belief State Template of ConvLab framework

2

Reinforcement Learning for Dialogue Management

In the previous chapter, we introduced the fully-operational Spoken Dialogue System (SDS) based on a complex pipeline including several sub-modules. The subject of our thesis focuses on one of these modules: the Dialogue Manager (DM). From now on, we consider the DM as an *agent* and we abstract all others DS modules and the user into a single entity called *environment*. We claim that the user as well as the communication modules of the dialogue system are fully integrated in the *environment* and we assume that the integrated modules play their role properly.

In this chapter, we discuss the main arguments for using RL in a dialogue application and present the mathematical framework for its modeling. Then, we lay the technical foundations for understanding RL (basic principles and ideas behind RL). Finally, we propose an overview of more sophisticated mechanisms used for DM with the RL paradigm.

2.1 Problem formulation of dialogue management

In the following, we present the mathematical modeling of RL applied in the context of task-oriented dialogues. Indeed, we claim that this modeling is adapted to the resolution of task-oriented dialogue because the basic characteristics of dialogue interactions are aligned with the principles behind RL. First the dialogue is *temporal* in the sense that the current dialogue interaction has an impact on how the dialogue progresses in the future. The dialogue is also *dynamic* in the sense that dialogues evolve differently according to the interlocutors - different reactions or ways of interacting - and according to the contexts - stochastic environments with different levels of noise (Rieser and Lemon, 2011). Thus DM requires foresight, long-term planning and robustness to deal with the dynamics of the environment. These considerations lead us naturally to the concept of Markov Decision Process that we will present.

2.1.1 Markov Decision Process framework

In mathematics, a **Markov Decision Process (MDP)** is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker (Bellman, 1957; Howard, 1960). We propose here to formalise it:

Definition 2.1: Markov Decision Process (MDP)

A **Markov Decision Process (MDP)** is a discrete-time stochastic control process and formally described by a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ where:

- $\mathcal{S}$ is a set of states called the *state space*,
- $\mathcal{A}$ is a set of actions called the *action space* (alternatively, $\mathcal{A}_s$ is the set of actions available from state s),
- $\mathcal{P} \in \Delta\mathcal{S}^{\mathcal{S}\times\mathcal{A}}$ is a probability function called *transition probability* where

$$\mathcal{P}(s'|s,a) = \mathcal{P}(S_{t+1} = s'|S_t = s, A_t = a) \tag{2.1}$$

  corresponds to the probability that action a in state s leads to state s',
- $\mathcal{R} \in \Delta\mathbb{R}^{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}$ is a probability function called *reward function* where

$$\mathcal{R}(r'|s,a,s') = \mathcal{R}(R_{t+1} = r'|S_t = s, A_t = a, S_{t+1} = s') \tag{2.2}$$

  corresponds to the probability that immediate reward r' will receive after transitioning from state s to state s', due to action a.

A MDP is modelling by a sequence of random variables $\{(S_t, R_t, A_t)\}_{0\leq t\leq T}$ and an outcome of a MDP is a trajectory $\tau = \{(s_t, r_t, a_t)\}_{0\leq t\leq T}$ of length T.

An **environment** is the engine of a MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$.

An **agent** is the entity interacting with an environment that is formally described by a mapping function $\pi \in \Delta\mathcal{A}^{\mathcal{S}} = \Pi$ from states to probabilities of selecting each possible action called *policy* and where

$$\pi(a_t|s_t) = \pi(A_t = a_t|S_t = s_t) \tag{2.3}$$

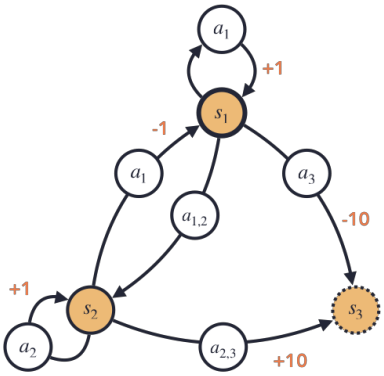
corresponds to the probability that action a_t in state s_t is selected.

Notation: If necessary, we use the tilde symbol ($\sim$) to denote "has the probability distribution of". In particular, we use the following notations:

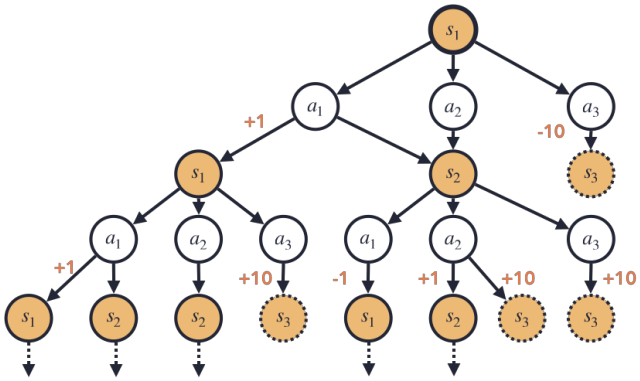
$$\pi(s) \sim \pi(\cdot|s)$$
$$r(s,a,s') \sim \mathcal{R}(\cdot|s,a,s')$$

If we develop the idea that a MDP is like a game, then this formalism is a way of modeling the point of view of a player called *agent* opposed to anything that does not depend on it called *environment*. The rules of the game governing the dynamics and the success and failure conditions are also external to the agent and thus included in the environment. What defines the agent is its strategy.

(a) Loop representation.



(b) Graph representation.



(c) Tree representation.

Figure 2.1: MDP representations. S_t, A_t and R_t are random variables. $\mathcal{S} = \{s_i\}_{1 \leq i \leq 3}$ and $\mathcal{A} = \{a_i\}_{1 \leq i \leq 3}$ are state and action spaces. Rewards are represented in orange.

In practice, the MDP is an iterative process in which an agent interacts with an environment over time. It proceeds as follows: at each time step t, the agent receives a state s_t in the state space $\mathcal{S}$ and executes an action a_t from the action space $\mathcal{A}$ according to its policy $\pi(a_t|s_t)$. Then the agent receives a reward signal $\mathcal{R}(s_t, a_t, s_{t+1})$ and transits to the next state s_{t+1} according to the transition probability $\mathcal{P}(s_{t+1}|a_t,s_t)$. Figure 2.1a represents this. Conceptually, a state s_t represents what the agent sees at time t. An action a_t represents what the agent does at time t. A reward r_{t+1} represents what the agent receives as feedback after performing the action a_t in state s_t. The agent strategy can be visualised as an agent travelling through a network of states interconnected by actions. In others words, the agent explores the state network by following the action paths (see Figure 2.1b for graph representation and Figure 2.1c for tree representation).

A MDP is a stochastic process that satisfies the Markov property: the evolution of the process in the future depends only on the present state and does not depend on past history, as expressed in Definition 2.2. Because the MDP is an extension of Markov Chain, the only difference is the addition of actions (allowing choice) and

rewards (giving motivation).

Definition 2.2: Markov Property

A MDP is represented by a sequence of random variables $(S_0, R_0, A_0), \ldots, (S_t, R_t, A_t)$ that respects the Markov property:

$$\mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = \mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t) \tag{2.4}$$

2.1.2 Dialogue modelling via Markov Decision Process

Depending on the nature and assumptions of the problem under consideration, a MDP can have several modellings (how to model the state space, the action space and the reward function, etc.). Without going into detail, we will explain how a dialogue problem is modelled via a MDP (and in particular via a POMDP) and what are its specificities (J. Williams, 2007; Young et al., 2013b).

From MDP to POMDP To begin with, it should be noted that the first proposed formalisation of dialogue problem was based on a MDP (Levin and Pieraccini, 1997). A more complete formalisation has been proposed the next decade by J. Williams and Young (2005) by considering than the dialogue system does not observe directly the user intent but its utterance. Therefore, the system have to infer what the user want to say. In practice, this function is managed by the DST module. That is why for the following we state that it is still possible to adapt a POMDP problem into a MDP problem without loss of generality.

Definition 2.3: Partial Observable Markov Decision Process

A **Partial Observable Markov Decision Process (POMDP)** is a discrete-time stochastic control process and formally described by a 6-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \mathcal{Z} \rangle$ where:

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ is a **MDP** (see Definition 2.1),
- $\mathcal{O}$ is a set of observations called the *observation space*,
- $\mathcal{Z} \in \Delta\mathcal{O}^{\mathcal{S}\times\mathcal{A}}$ is a probability function called *observation probability* where:

$$\mathcal{Z}(o|s, a) = \mathcal{Z}(O_{t+1} = o|S_{t+1} = s, A_t = a) \tag{2.5}$$

  corresponds to the probability that a previous action a that leads to the state s generates the observation o.

In this framework, at each time, the **environment** communicates an observation o instead of a state s to the **agent** from which it must infer its *belief state* $\tilde{s}$, the most likely state from its point of view.

State space The state space refers to the set of reachable states for the agent in the MDP. We called them **frames** (see Subsection 1.3.2), which are a knowledge structure (in practice a dictionary of probabilistic or deterministic distribution of slots, values and user/system actions) representing the history of the intentions that the systems have extracted from the sentences of the user and the system since the beginning of the conversation.

The input to the DM is the *belief state* that comes from the **master state space**. Due to its large size, this representation is projected into the **summary state space** by a process called *value abstraction* (see Figure 2.2). Finally, it must be vectorised in order to be interpretable by machine learning methods. We call the function that performs these different operations the *feature function* ϕ.

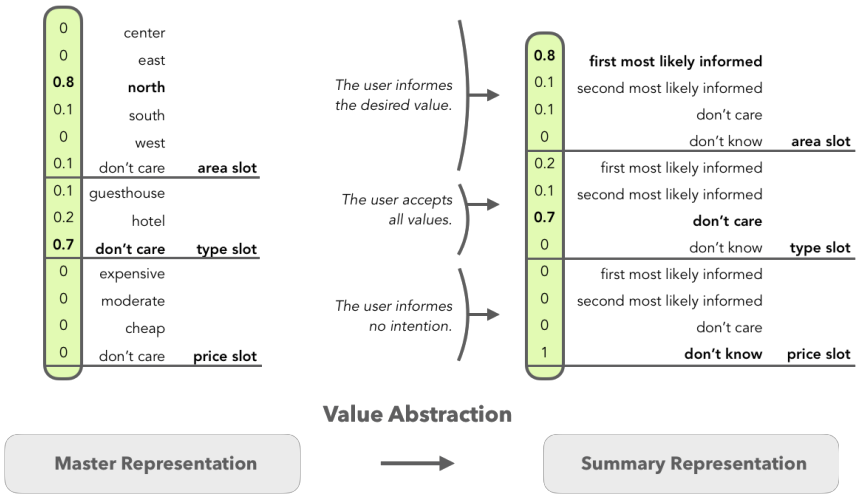


Figure 2.2: Processing the belief state from the master representation to the summary representation via *value abstraction*. Each slot is abstracted by keeping the probabilities of the first and second most likely informed values, as well as the probabilities of the "don't care" and "don't know" values (equal to 1 if no value was informed). By following this process, the summary representation is independent of the number of values in each slot.

In the case of task-oriented dialogue interactions, this space is **finite** because it is delimited by the set of possible slots, values and actions of the domain ontology. But it is **continuous** in the sense that the data representation is not categorical (listing and tracking all possible combinations of these elements is not reasonable when the sets of possible slots, values and actions become intractable). See Table 1.5 and Table 1.6 in Chapter 1 for more details about how dialogue states are represented and Table 1.2 on ontology structure.

Action space The action space refers to the set of actions that can be taken by the agent. We called them **dialogue actions** (see Subsection 1.3.2) which are formalised as predicates, possibly with slot or slot-values pairs as arguments. Therefore the agent's policy is a probabilistic distribution over these actions and it selects only one action per turn.

To reduce the complexity of the learning problem, **master actions** which are valued dialogue acts such as INFORM(date = "2022-01-15") are abstracted into **summary actions** like INFORM(date), the *value abstraction* module being in charge of restoring the relevant values in the context. Figure 2.3 illustrates this process.

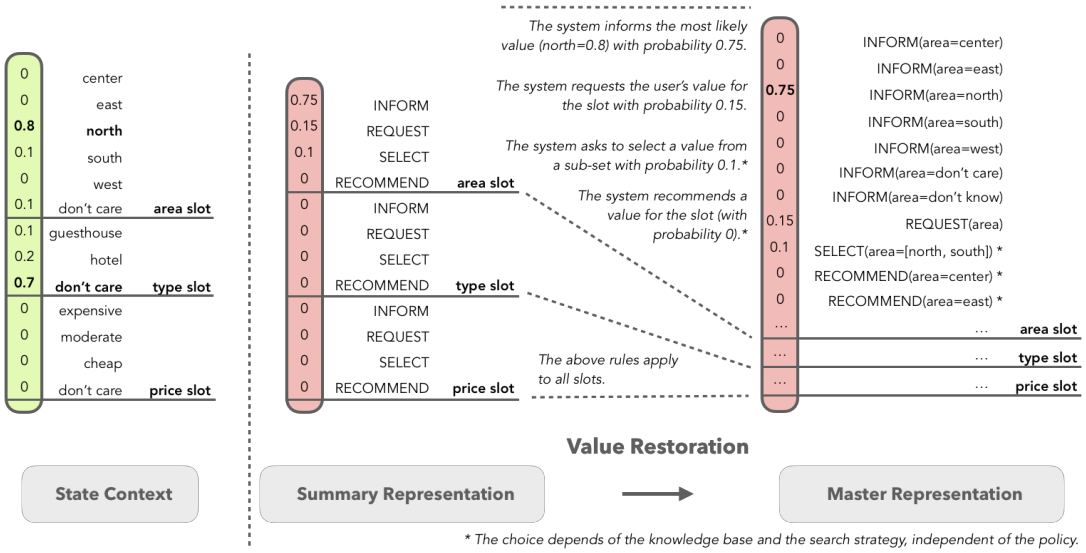


Figure 2.3: Processing the probabilistic action (red columns) from the summary representation to the master representation via *value restoration* that depends on the state context (green column). The INFORM action informs of the most likely value of the slot concerned. The REQUEST action does not need to be linked to a value. The SELECT and RECOMMEND actions use hand-crafted control to choose the value to be communicated. At the end of this process, the master representation is restored according to the ontology.

The **master action space** is **continuous** in the sense that the size of values is unknown a priori but the **summary action space** is **discrete** because it is limited by the number of possible predicates and slots of domain ontologies.

On the PyDial platform, the actions are performed one by one (called *single-actions*). On ConvLab, the policy may activate several actions simultaneously (called *multiple-actions*). On both systems, some hand-crafted control may be added through a mask of valid actions in order to disable not pertinent actions depending of the context (*i.e.* in the beginning or at the end of the conversation). Figure 2.4 summarises the entire decision-making process, from belief state to dialogue acts.

Due to structural limitations of our future approaches and in order to simplify our problem by limiting the number of actions at first, we restrict our agent to choose only *single-actions*. Subsequently, we will specify whether this condition will be lifted. If necessary, in order to cope with ConvLab's multiple-actions, we encode them as single-actions by defining additional macro-actions like the MAKE_OFFER action that finds the relevant slots to inform, or the MAKE_RESERVATION action that finds the relevant item to book and to inform.

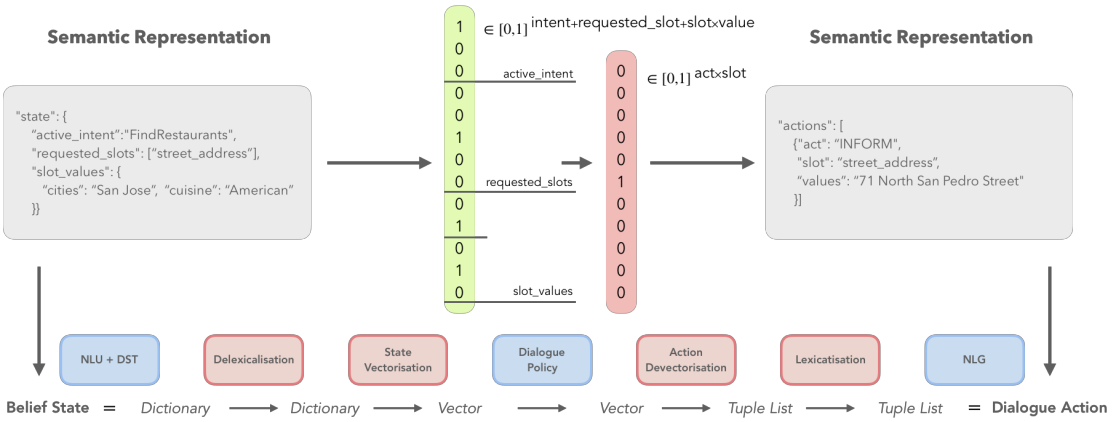


Figure 2.4: State/action representation and transformation process for dialogue system. The semantic representation of the state is transformed from dictionary to vector. The decision is then transformed from vector to dialogue action.

Time horizon & Reward function In task-oriented dialogues, the conversation begins with greetings and ends systematically whether it was a success or a failure. A conversation is time-bound and is therefore **episodic** (vs. continuous in time). Abusively, we note T the time horizon of the conversation episodes, *i.e.* the maximum number of dialogue turns allowed for the agent to achieved its objective.

The objective of the agent is not well defined because it depends on the reward function that is not explicit (contrary to video games or board games). Objectively, we can expect that the agent achieves a maximum number of conversations (therefore a maximum number of episodes) in a minimum number of dialogue turns. In practice, we reward the agent at the end of an episode if the dialogue is a success and we penalise it at each time step to prevent long dialogues.

Reward function in DM problem Single-domain dialogue framework:

$$\mathcal{R}(s', a, s) = \begin{cases} r_M = 20, & \text{if } s' \text{ is a terminal state of a successful dialogue.} \\ p = -1, & \text{else.} \end{cases} \tag{2.6}$$

Multi-domain dialogue framework:

$$\mathcal{R}(s', a, s) = \begin{cases} r_M = 40, & \text{if all domains in } s' \text{ are solved.} \\ r_m = 5, & \text{if the current active domain in } s' \text{ is solved.} \\ p = -1, & \text{else.} \end{cases} \tag{2.7}$$

where r_M is a major reward ($r_M = T$), r_m is a minor reward ($0 < r_m < T$) and p is a penalty ($p = -1$).
A domain is solved if all related tasks are solved.

2.1.3 The reinforcement learning problem

After presenting the core concept of MDP, we understand that a MDP model gives us a context for designing the agent's policy. It enables us to formalise a MDP as a mathematical optimisation problem which can be solved using Reinforcement Learning (R. Sutton and Barto, 2018). We will define the objective of the agent.

First, the idea behind a MDP is to discover the control strategy that lead to an optimal behaviour with respect to a criteria: the **reward**. Indeed, the reward is as a motivation signal for the agent in order to estimate the quality of its actions. So we naturally define the concept of **return**.

Definition 2.4: Return (or cumulative reward)

Let $\gamma \in [0,1[$ be the *discount rate* that determines the present value of future rewards. Then the **return** at time t is defined as the sum of future discounted rewards. We call g_t the *cumulative reward for a trajectory* τ and G_t the *cumulative reward for a policy* π:

$$g_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \qquad G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{2.8}$$

where T is the horizon, $R_i \sim \mathcal{R}(R_i|S_{i-1}, A_{i-1}, S_i)$, $S_i \sim \mathcal{P}(S_i|S_{i-1}, A_{i-1})$, $A_i \sim \pi(A_i|S_i)$ and $S_0 \sim \mathcal{P}(S_0)$.

According to this definition, the agent can estimate the quality of its trajectories and of its policy. Its objective is so to have the best policy in estimation, so the optimal policy according to the dynamics of the environment.

Definition 2.5: Objective

Let $\gamma \in [0,1[$, then the **objective** of an agent in an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ consists of finding an **optimal policy** π^* that maximises the *return* in expectation:

$$\pi^* \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{P},\pi}[G_0] = \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{k=0}^{T} \gamma^k R_{k+1}\right] \tag{2.9}$$

Where a MDP provides a framework for the formalisation of the dialogue problem, **reinforcement learning** is one of the tools for its solution (as we will see later in Section 2.4). Among the other solutions that have been commonly used, the **hand-crafted policy** and **learning from demonstration** approaches are alternative solutions to the same problem based on different assumptions with their strengths and weaknesses. First we propose a brief overview on these approaches respectively in Section 2.2 and Section 2.3.

2.2 Hand-crafted policy approaches

Hand-crafted approaches rely on programs or/and models that are fully specified by developers or domain experts to track the dialogue state and define the policy. By this process, experts are considered to know the optimal policy (or to a lesser extent the closest to it) and to be able to render it in the form of explicit algorithms. Following the review of DM approaches proposed by Brabra et al., 2021, it is possible to identify four kinds of hand-crafted approaches:

Rule-based approach in which bot developers define the dialogue state as well as the policy by encoding a set of rules. The simplest modeling of these rules is structuring them in the form of pattern/response pairs, which perform NLU, DM, and NLG tasks at once by taking the user utterance and producing the corresponding response.

Finite state machine-based approach that is a declarative approach providing a set of a predefined sequence of steps representing the dialogue state at any point during the conversation. Each state is restricted to a prescribed number of transitions to other states and defines the set of actions that the conversational system can/should perform in a given situation.

Activity-based approach that is a procedural approach providing a concrete implementation of DM by specifying the workflow that it may go through during the conversation. It allows the development of DM model by leveraging the concept of workflows, namely activities, triggers and actions.

Frame-based approach that relies on a domain ontology that defines a set of frames, each specifying the required information that the conversational system is designed to acquire from the user in order to fulfill a dialogue task.

The frame-based approach to conversation systems offers greater flexibility in handling over-informative inputs from users by allowing them to fill in slots in various orders and combinations. However, this approach necessitates the use of advanced algorithms to determine the next system action or question, based on a set of features including the user's previous utterance, the remaining slots to be filled, and dialogue control priorities. Additionally, thorough testing is required to ensure that the system does not ask inappropriate questions in unforeseen situations. Despite these challenges, the frame-based approach remains a prevalent method in modern conversation systems, such as Apple's Siri, Amazon's Alexa, and Google Assistant, Furthermore, it is implemented in current dialogue frameworks such as PyDial and ConvLab.

2.3 Learning from demonstrations approaches

In contrast to hand-crafted approaches where the DM logic has to be defined by hand, data-driven approaches were proposed to learn the dialogue policy from demonstrations, namely **Learning from Demonstrations (LfD)** by which an apprentice agent learns a control policy for an environment by observing demonstrations delivered by an expert agent. It can be viewed as a form of supervised learning, where the training data set consists of task executions by a demonstration teacher (Argall et al., 2009). It is usually implemented as either **Imitation Learning (IL)** or **Inverse Reinforcement Learning (IRL)**.

Whereas IL is a learning paradigm that consists in generalising an (assumed optimal) expert policy π_E based on expert demonstrations D_E, IRL is a learning paradigm that consists in finding an (assumed non trivial) expert reward function $\mathcal{R}_E$ based on expert demonstrations D_E that could explain its behaviour. It has been shown that these two paradigms are equivalent in the sense that there exists an explicit bijective operator between their respective solution spaces, within the context of the set-policy framework (Piot et al., 2017).

2.3.1 Imitation Learning

Definition 2.6: Imitation Learning (IL)

Imitation Learning (IL) refers to methods that aim to generalise the expert policy π_E, as observed in the expert data set D_E, to new situations. To be precised, IL methods learn from the expert's actions and try to replicate them in unseen states.

Batch score-based IL is often reduced to Multi-class Classification (MCC) by minimising the following empirical risk:

$$\text{Given} \quad D_E = (s_k, a_k)_{1 \leq k \leq N}, \qquad \pi = \arg\min_{\pi \in \Pi} \frac{1}{N} \sum_{(s_k, a_k) \in D_E} \mathbf{1}_{\{\pi(s_k) \neq a_k\}} \tag{2.10}$$

In practice, the goal of IL method is to compute a score function $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ that gives the highest scores to at least some of expert actions:

$$\forall s \in \mathcal{S}, \quad \arg\max_{a \in \mathcal{A}} Q(s, a) \subset \text{Supp}(\pi_E(\cdot|s)) \tag{2.11}$$

The decision rule π associated to the score function Q consists in taking the action with the best score:

$$\forall s \in \mathcal{S}, \quad \pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a) \tag{2.12}$$

In practice, minimising the empirical risk as defined in Eq. 2.10 is computationally difficult, and practitioners often resort to using convex surrogates to approximate the true risk. Additionally, it can be observed that the empirical risk does not take into account the dynamics of the MDP. As a result, the dynamic nature of the environment is ignored, which is a significant drawback of IL. Despite this limitation, MCC methods are relatively easy to implement, and have been theoretically justified in both finite and infinite horizon settings.

Algorithms of IL: Standard IL methods use expert demonstrations to guide the agent in learning relevant situations and expert corrections to guide the agent to recover from past mistakes. Among the most popular, SEARN (Daumé et al., 2009) is one such method, which maintains a current policy and uses it to generate new training data to learn a new policy. SMILe (Stephane Ross et al., 2010) is another method that builds on SEARN, with a simpler implementation and less expert interaction. DAgger (Stéphane Ross et al., 2011) learns a policy that mimics the expert policy on states induced by previous learned policies, with the expert providing corrections during exploration to help the agent recover from past mistakes. AggraVate (Stéphane Ross et al., 2014) is an extension of DAgger, which learns to choose actions to minimise the cost-to-go of the expert rather than mimicking their actions. LOLS (Chang et al., 2015) is a learning-to-search (L2S) algorithm that has been shown to have superior performance compared to other L2S algorithms in cases where the reference policy performs poorly, but local hill-climbing in policy space is effective.

Combined IL and RL: Hybrid imitation and reinforcement learning refers to a subset of IL methods where the agent has been trained on expert demonstrations and on its own interactions with the environment to generate more trajectories guided by rewards. There are different ways to integrate IL into the RL process: (i) using IL as a pre-training phase, also known as Behaviour Cloning (BC) (Su et al., 2016a; Liu et al., 2018; Y. Gao et al., 2019; Goecks et al., 2020); (ii) using RL with only demonstrations as a pre-training phase (Hester et al., 2017; Y. Gao et al., 2019; Goecks et al., 2020); (iii) using both SL and RL losses during the RL phase (Hester et al., 2017; Goecks et al., 2020; Gordon-Hall et al., 2020). More recent methods use non-standard ways to use expert demonstrations, such as learning from a single demonstration with resetting to a demo state (Salimans et al., 2018), learning a weak and cheap expert policy for simulation (Gordon-Hall et al., 2020) or computing intrinsic motivation from demonstrations (Hussenot et al., 2021). To complement this overview, surveys of IL methods has been proposed by Hussein et al. (2017) and Attia et al. (2018).

2.3.2 Inverse Reinforcement Learning

Definition 2.7: Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning (IRL) refers to methods that aim to find a reward function $\mathcal{R}$ that can explain the expert policy π_E. The expert policy is assumed to be optimal with respect to an unknown reward function $\mathcal{R}_E$. IRL methods use expert demonstrations to infer this reward function and understand the expert's decision-making process.

In order to quantify the quality of a policy π with respect to the reward $\mathcal{R}$, the *quality function* $Q^\pi_\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, also called the state-action value function (see Subsection 2.4.1), is defined. The function $Q^*_\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is called the *optimal quality function* and is defined as:

$$Q^*_\mathcal{R} = \max_{\pi\in\Pi} Q^\pi_\mathcal{R} \tag{2.13}$$

The goal of an IRL algorithm is to compute a non-trivial reward function $\mathcal{R}$ for which at least a subset of expert actions and only expert actions are optimal to prevent non-expert actions to be optimal:

$$\forall s \in \mathcal{S}, \quad \arg\max_{a\in\mathcal{A}} Q^*_\mathcal{R}(s,a) \subset \text{Supp}(\pi_E(\cdot|s)) \tag{2.14}$$

Once the reward function is learned, the MDP must be solved with RL (see Section 2.4) to compute the apprentice policy π such as:

$$\forall s \in \mathcal{S}, \quad \pi(s) = \arg\max_{a\in\mathcal{A}} Q^*_\mathcal{R}(s,a) \tag{2.15}$$

IRL is a challenging task because the expert's actions may be influenced by multiple factors. The expert's reward function may be complex and non-linear. IRL methods have been developed in various forms, starting from the early work of Ng and Russell (2000) who proposed the first algorithm for IRL based on linear programming and continuing with the work of Abbeel and Ng (2004) who proposed an alternative version with quadratic programming. Since then, many other methods have been proposed, such as Bayesian IRL (Ramachandran, 2007; Michini et al., 2012), maximum entropy IRL (Ziebart et al., 2008) and non-linear IRL with gaussian processes (Levine et al., 2011). Other algorithms developed during the same period have been presented in surveys (Zhifei et al., 2012; Ghavamzadeh et al., 2015). More recently, deep IRL and more sophisticated methods have been proposed which use deep neural networks to model the reward function such as maximum entropy deep IRL (Wulfmeier et al., 2016), adversarial IRL (Fu et al., 2018) (inspired by generative adversarial IL by Ho et al. (2016)), Meta-IRL with probabilistic context variables (Yu et al., 2019) and IRL as hindsight relabeling (Eysenbach et al., 2020).

IRL for dialogue systems: IRL has also been applied to task-oriented dialogue problems, where the objective is to infer the underlying reward function of a human user by understanding their preferences. One of the first works in this area was presented by Boularias et al. (2010), who proposed an IRL algorithm for learning the reward function of a dialogue model from human-human dialogues. Chandramohan et al. (2011a) and Chandramohan et al. (2012) proposed a novel approach using IRL for building multiple behavior-specific user simulators using IRL. Pietquin (2013) discussed a method for learning a reward function for human-machine interaction using Bayesian IRL. Rojas-Barahona et al. (2014) proposed a Bayesian approach for learning the behaviour of human characters using Bayesian IRL in a serious game context. Furthermore, there are various IRL methods for using expert demonstrations that can be broadly divided into two categories: (i) learning a reward function directly (Chandramohan et al., 2011a; Rojas-Barahona et al., 2014), by using deep adversarial method (Fu et al., 2018) or joint reward learning and policy optimisation (Takanobu et al., 2019a; X. Huang et al., 2020) (ii) engineered reward or engineered policy, such as expert-based reward shaping and exploration schemes (Ferreira, Emmanuel et al., 2013), social signal and user adaptation (Ferreira et al., 2013) or reward and policy shaping (H. Wang et al., 2020). Finally, some methods propose to learn directly with human teaching and feedback instead of tuned reward signal (Liu et al., 2018).

2.4 Reinforcement learning approaches

Reinforcement Learning (RL) is a type of machine learning that differs from supervised and unsupervised learning, in which an agent learns to make decisions by interacting with its environment and receiving feedback in the form of rewards. The agent's goal is to learn a policy, which maps states of the environment to actions, that maximises the cumulative reward over time. RL algorithms typically use value functions (see Definition 2.8) to estimate the long-term reward of different actions in different states, and use this information to update the policy.

RL can be used to solve a wide range of tasks, including games, robotics, decision making problems and, particularly in our case, task-oriented dialogue problems. As RL methods are commonly used in our field of research, we propose a summary to introduce the basic elements needed to understand the ins and outs of our work and explain the challenges we face when implementing algorithms in RL. For an enriched and complete introduction to the subject, we invite the interested reader to consult the work of R. Sutton and Barto (2018) and the work of Y. Li (2018) which provide a broad overview of the state of the art in RL.

2.4.1 Theoretical foundation and core elements

Value functions

As explained earlier, the core of RL algorithms is based on value functions that estimate the long-term reward of different actions in different states.

Definition 2.8: Value Functions

The **value function** of a state s under a policy π, denoted $v^\pi(s)$, is the expected return when starting in s and following π thereafter. We call v^π the *state-value function for policy* π:

$$\begin{aligned} \forall s \in \mathcal{S}, \quad v^\pi(s) &= \mathbb{E}_{\mathcal{P},\pi}\left[G_t | S_t = s\right] \\ &= \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s\right] \end{aligned} \tag{2.16}$$

The **value function** of taking action a in state s under a policy π, denoted $q^\pi(s,a)$, is the expected return starting from s, taking the action a, and following π thereafter. We call q^π the *state-action-value function for policy* π:

$$\begin{aligned} \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad q^\pi(s,a) &= \mathbb{E}_{\mathcal{P},\pi}\left[G_t | S_t = s, A_t = a\right] \\ &= \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \end{aligned} \tag{2.17}$$

Partial Ordering and Optimality

Solving a reinforcement learning task means, roughly, finding a policy that maximises long-term reward. Value functions provide a partial ordering over policies, where a policy is considered better or equal to another if it has greater or equal expected return for all states. An optimal policy, denoted by π^* (there may be multiple optimal policies) is one that is better than or equal to all other policies and shares the same state-value function v^* and state-action-value function q^*.

Theorem 2.1: Optimal Value Functions and Optimal Policy

The **values functions** are called **optimal** if and only if:

$$\begin{aligned} \forall s \in \mathcal{S}, \quad \forall \pi \in \Pi, \quad v^*(s) \geq v^\pi(s) \\ \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \forall \pi \in \Pi, \quad q^*(s,a) \geq q^\pi(s,a) \end{aligned} \tag{2.18}$$

In that respect, a **policy** is called **optimal** if and only if:

$$\begin{aligned} \forall s \in \mathcal{S}, \quad \pi^* \in \arg\max_{\pi \in \Pi} v^\pi(s) \\ \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \pi^* \in \arg\max_{\pi \in \Pi} q^\pi(s,a) \end{aligned} \tag{2.19}$$

Recursive Relationships

A fundamental property of value functions used throughout RL is that they satisfy recursive relationships: the **Bellman Equation** (as presenting in Theorem 2.2 and Theorem 2.3). describes the value of a state (or a state-action pair) in terms of the value of the next state (or the next state-action pair) and the rewards that will be received. It provides a way to iteratively update the value function of a policy.

Theorem 2.2: Bellman Expectation Equations

Bellman Expectation Equation for v^π function:

$$\begin{aligned} v^\pi(s) &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \ [r(s,a,s') + \gamma \, v^\pi(s')] \\ &= \mathbb{E}_{\mathcal{P},\pi} \left[R_{t+1} + \gamma \, v^\pi(S_{t+1}) | S_t = s \right] \end{aligned} \tag{2.20}$$

Bellman Expectation Equation for q^π function:

$$\begin{aligned} q^\pi(s,a) &= \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left[r(s,a,s') + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') \, q^\pi(s',a') \right] \\ &= \mathbb{E}_{\mathcal{P},\pi} \left[R_{t+1} + \gamma \, q^\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a \right] \end{aligned} \tag{2.21}$$

Theorem 2.3: Bellman Optimality Equation

Bellman Optimality Equation for v^* function:

$$\begin{aligned} v^*(s) &= \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left[r(s,a,s') + \gamma \, v^*(s') \right] \\ &= \mathbb{E}_{\mathcal{P},\pi^*} \left[R_{t+1} + \gamma \, v^*(S_{t+1}) | S_t = s \right] \end{aligned} \tag{2.22}$$

Bellman Optimality Equation for q^* function:

$$\begin{aligned} q^*(s,a) &= \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}(s')} q^*(s',a') \right] \\ &= \mathbb{E}_{\mathcal{P},\pi^*} \left[R_{t+1} + \gamma \, q^*(S_{t+1}, A_{t+1}) | S_t = s, A_t = a \right] \end{aligned} \tag{2.23}$$

In essence, finding an optimal policy by solving the Bellman optimality equation requires knowledge of and access to the dynamics of the environment, assumptions that can hardly be met in a real application.

Policy Improvement Theorem

The above considerations give rise to an important property that is the basis of all RL methods: the **Policy Improvement Theorem** (presented in Theorem 2.4) is a fundamental result that states that for any given policy, there exists another policy that is at least as good and possibly better. In other words, it guarantees that we can always improve the current policy by following the greedy policy with respect to the current action-value function (Watkins, 1989).

Theorem 2.4: Policy Improvement Theorem

Let π and π' be policies and let π' be chosen so that:

$$\forall s \in \mathcal{S}, \quad q^{\pi}(s, \pi'(s)) \geq v^{\pi}(s) \tag{2.24}$$

Then it follows that π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states:

$$\forall s \in \mathcal{S}, \quad v^{\pi'}(s) \geq v^{\pi}(s) \tag{2.25}$$

From this result, it is possible to propose algorithms that find an optimal policy: the algorithm called **Value Iteration** (VI) can find an optimal value function by iterating on Equation 2.23 (Bellman, 1957); and the **Policy Iteration** algorithm (PI) can find an optimal policy by evaluating the current policy with Equation 2.21 then by improving it greedily over the current value function of the policy with Equation 2.24 (Howard, 1960).

Generalised Policy Iteration

In what follows, we will consider that any standard RL algorithm can be seen as a generalised policy iteration algorithm that is a unified framework that encompasses the classical PI and VI algorithms.

Definition 2.9: Generalised Policy Iteration (GPI)

A **Generalised Policy Iteration (GPI)** is a framework for solving RL problems that combines both policy evaluation and policy improvement that can be performed simultaneously and interactively. It consists of two steps:

- **Evaluation step**: where the value function is evaluated for the current policy, usually via the Bellman Equation.
- **Improvement step**: where a next policy is obtained from the current value function via the Policy Improvement Theorem, usually via a greedy strategy.

The general mechanism of most RL algorithms is as follows: at the initialisation step, the agent starts with an initial Q-value function, denoted $\hat{q}(s,a)$, and policy function, denoted $\hat{\pi}(s)$, where all the state-action values are set to some arbitrary value. In the evaluation step, the agent updates its expected Q-values incrementally using the Bellman Equations, which can be represented as:

$$\hat{q}_{new}(s,a) \leftarrow \hat{q}_{old}(s,a) + \alpha_{step}[q_{target}(s,a) - \hat{q}_{old}(s,a)]$$

where α_{step} is the step size that controls changes over time, q_{target} is a sampled Q-value and $[q_{target}(s,a) - \hat{q}_{old}(s,a)]$ can be seen as the error toward the optimal value q^*. In the improvement step, the agent improves its policy by following a greedy strategy, *i.e.*:

$$\hat{\pi}_{new}(s) \leftarrow \arg\max_{a \in \mathcal{A}} \hat{q}_{new}(s,a)$$

2.4.2 Classical Reinforcement Learning Algorithms

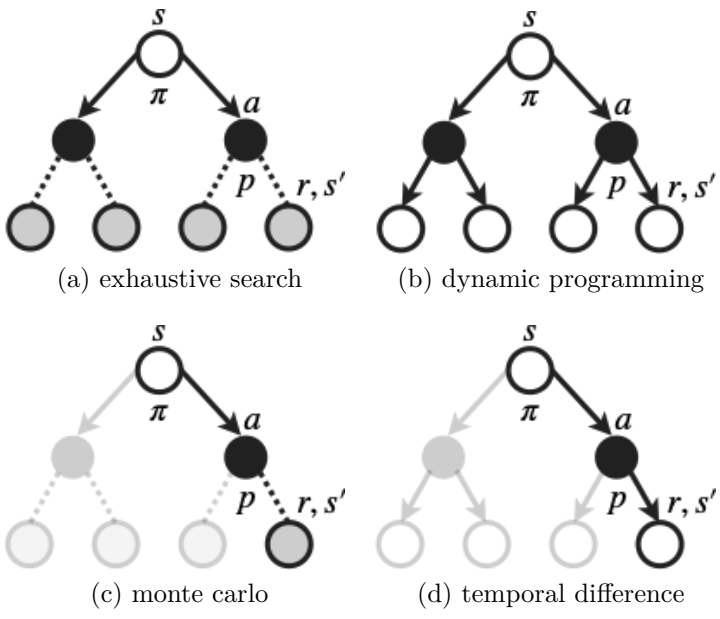
In practice, the RL algorithms seen through the prism of GPI must learn a value and policy functions and periodically evaluate and improve them. We distinguish between three types of methods: (i) **value-based methods** that are based on learning a value function, (ii) **policy-based methods** that learn the policy function directly, and (iii) **actor-critic based methods** that optimise both the value function and a policy function. We briefly provide an overview of these classical approaches which are the basis for the more sophisticated state-of-the-art algorithms.

Learning a value function

The methods for learning the value function differ according to the nature of the updates performed (depth and/or width updates). Furthermore, the main difference between Dynamic Programming (DP) methods and Temporal Difference (TD) methods lies in the opposition between model-based and simulation-based approaches. Secondly, both approaches use bootstrapping (updating targets that include existing estimates) to evaluate the target Q-value whereas the Monte Carlo (MC) methods do not. For the equations, we only express the state-action-value function q^π.

Dynamic Programming methods: They consist of using the Bellman Equation to update the value function (bootstrapping with width updates). They assume knowledge of an exact mathematical model of the MDP and therefore become infeasible for large MDPs. Formally, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$q^\pi(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left[r(s',a,s) + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s')\, q^\pi(s',a') \right] \tag{2.26}$$



(a) exhaustive search (b) dynamic programming

(c) monte carlo (d) temporal difference

Figure 2.5: Diagrams illustrating the procedure for updating the state-value function v^π (works as well for q^π) according to the different methods presented. A white circle represents a state, a black circle an action and a grey circle a terminal state. An arrow represents a transition from a state to an action according to the policy π and from an action to the next state and the reward according to the dynamics of the environment p. The paths of the tree in transparency are not explored by the methods and those in dotted line are explored in depth.

Monte Carlo methods Model-free methods that update the value function directly from experience, based on averaging complete returns (sampling with depth updates with low bias but high variance). For each $\tau = \{(s_t, r_t, a_t)\}_{0 \leq t \leq T}$:

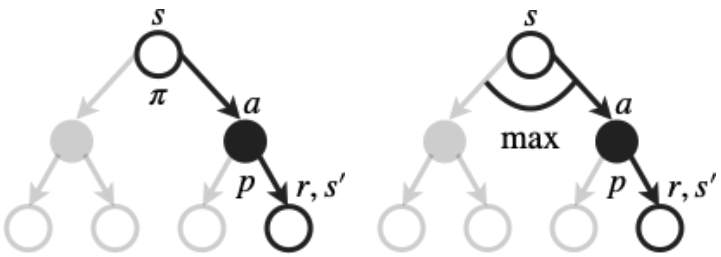
$$q^\pi(s_t, a_t) \leftarrow q^\pi(s_t, a_t) + \alpha \left[g_t - q^\pi(s_t, a_t) \right] \tag{2.27}$$

with $\alpha = 1/n$ when n is the number time when (s_t, a_t) is met and g_t is the cumulative return of the trajectory τ at time t (see Eq. 2.8).

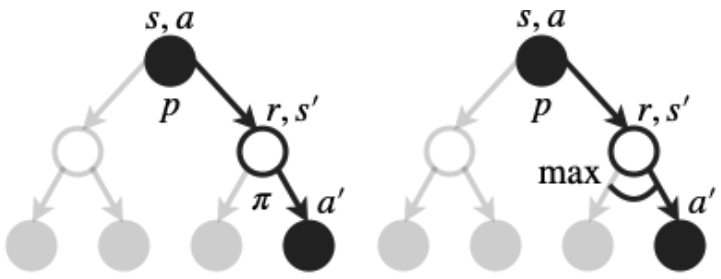
Temporal Difference methods Model-free methods that update the value function directly from experience, based on accumulating partial returns (bootstrapping with one-step sampled updates with high bias but low variance). The most popular algorithms in this category are TD learning (R. Sutton, 1988), SARSA (Rummery et al., 1994) (in Eq 2.28) and Q-learning (Watkins, 1989) (in Eq 2.29). For each $\tau = \{(s_t, r_t, a_t)\}_{0 \leq t \leq T}$:

$$q^\pi(s_t, a_t) \leftarrow q^\pi(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \, q^\pi(s_{t+1}, a_{t+1}) - q^\pi(s_t, a_t) \right] \tag{2.28}$$

$$q^\pi(s_t, a_t) \leftarrow q^\pi(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a q^\pi(s_{t+1}, a) - q^\pi(s_t, a_t) \right] \tag{2.29}$$



(a) TD (v^π) (b) Optimal TD (v^*)



(c) SARSA (q^π) (d) Q-Learning (q^*)

Figure 2.6: Diagrams illustrating the procedure for updating the value function according to TD methods. A white circle represents a state and a black circle an action. An arrow represents a transition from a state to an action according to the policy π (or according to the greedy strategy max) and from an action to the next state and the reward according to the dynamics of the environment p. The paths of the tree in transparency are not explored by the methods.

Learning a value function with function approximation

We discuss tabular cases above, where a value function is stored in a tabular form. Function approximation is a way for generalisation when the state and/or action spaces are large or continuous. Linear approximation of the function was a popular choice before the deep Q-network work of Mnih et al. (2015) marked the emergence of deep learning in RL.

Value function approximation The gradient of a value function q^π parametrised by a model ω is computed as follows:

$$\begin{aligned}\nabla_\omega q^\pi &= \nabla_\omega \mathrm{E}_{\mathcal{P},\pi}\left[(q^\pi(S_t, A_t; \omega) - \hat{q}(S_t, A_t))^2\right] \\ &\propto \mathrm{E}_{\mathcal{P},\pi}\left[(q^\pi(S_t, A_t; \omega) - \hat{q}(S_t, A_t))\nabla_\omega q^\pi(S_t, A_t; \omega)\right]\end{aligned} \tag{2.30}$$

where $\hat{q}$ is the current estimator of q^π defined according to the value function learning method used (TD, SARSA, Q-learning, etc).

Learning a policy function

In contrast to value-based methods, policy-based methods optimise the policy directly with function approximation and update the parameters by gradient ascent (see Theorem 2.5 that gives a formulation of the policy gradient *w.r.t* the value function). They are known for their superior convergence characteristics and ability to handle high-dimensional or continuous action spaces, and to learn stochastic policies. However, they tend to converge to a local optimum, are computationally less efficient, and have higher variance in the estimates (Y. Li, 2018).

Theorem 2.5: Policy Gradient Theorem

The gradient of a policy function π parametrised by a model θ is computed as follows (R. Sutton et al., 2000):

$$\nabla_\theta \pi = \nabla_\theta \mathbb{E}_{\mathcal{P},\pi}[G_0] \propto \mathrm{E}_{\mathcal{P},\pi}\left[Q^\pi(S_t, A_t)\, \nabla_\theta \log \pi(A_t|S_t; \theta)\right] \tag{2.31}$$

where G_0 is the initial return and Q^π is any state-action value estimate, possibly with a comparison to an arbitrary baseline that does not vary with action.

REINFORCE algorithm (R. Williams, 1992) is a policy-based method that calculates the expected return as an estimate of the state-action value as $Q^\pi(s_t, a_t) = g_t$ involving $\nabla_\theta \pi \propto \mathrm{E}_{\mathcal{P},\pi}\left[G_t\, \nabla_\theta \log \pi(A_t|S_t; \theta)\right]$.
Actor-Critic algorithms are policy and value-based methods that optimise both policy and value functions: $Q^\pi(s_t, a_t) = \hat{q}(s_t, a_t; \omega)$ for Q actor-critic, $Q^\pi(s_t, a_t) = \hat{q}(s_t, a_t; \omega) - \hat{v}(s_t; \omega)$ for advantage actor-critic and $Q^\pi(s_t, a_t) = r_t + \gamma \hat{v}(s_{t+1}; \omega) - \hat{v}(s_t; \omega)$ for TD actor-critic.

2.4.3 Advanced mechanisms for dialogue management

In the world of RL, algorithms are multiplying year after year offering more and more sophisticated techniques for computational and data efficiency. The emergence of deep learning (LeCun et al., 2015) has led to a significant growth in RL methods, also called Deep Reinforcement Learning (DRL) (Y. Li, 2018), that have not failed to influence the field of task-oriented dialogue systems (J. Gao et al., 2018).

Deep Reinforcement Learning

Among the value-based methods, the Deep Q-Network (DQN) and Double DQN algorithms presented by Mnih et al. (2013) and Hasselt et al. (2015) use deep neural networks to represent the Q-values of the state-action pairs. Regarding the actor-critic methods, the Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al., 2016) uses a parallel and asynchronous approach to update the actor and critic models. Ziyu Wang et al. (2017) propose the Actor Critic with Experience Replay (ACER) algorithm, which combines several techniques to achieve stability and sample efficiency, including truncated importance sampling, stochastic dueling networks and trust region policy optimisation. De facto, trust region methods are approaches to stabilising optimisation by constraining gradient updates. The Trust Region Policy Optimization (TRPO) algorithm (Schulman et al., 2017a) is an iterative procedure to monotonically improve policies. The Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017b) alternates between data sampling and optimisation, and benefits from the reliability of TRPO, with the goal of simpler implementation, better generalisation, and better empirical sampling complexity.

DRL for dialogue management

Different policy learning methods have already been tested for task-oriented dialogue (Casanueva et al., 2017; Takanobu et al., 2020). To name but a few, KTD framework that is based on Kalman filters and Temporal Differences (Geist et al., 2010), was applied for sample-efficiency, non-linear approximation, non-stationarity handling and uncertainty management (Daubigney et al., 2012; Ferreira et al., 2013). In continuity, GP-SARSA (Gasic et al., 2013) is an on-line RL algorithm that incorporates gaussian processes regression, a non-parametric Bayesian model used for function approximation (Engel et al., 2005). Since then, DRL approaches as DQN have become the new baseline for task-oriented dialogue systems and are used in the majority of recent papers. Then, a variety of advanced deep approaches with actor-critic have been used to improve performance such as deep actor-critic methods with and without pre-supervised learning (Fatemi et al., 2016), Trust Region Actor Critic with Experience Replay (TRACER) methods with and without pre-supervised learning (Su et al., 2017), TRACER in large action spaces context (Weisz et al., 2018) and TRACER with intrinsic curiosity module (Wesselmann et al., 2019).

Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) is an approach of reinforcement learning where the learning process is decomposed into multiple levels or stages, in order to learn complex tasks by breaking them down into smaller subtasks and learning to solve each one independently before combining them into a larger solution. There are several different approaches to HRL, including feudal learning framework, options framework, MAXQ framework and hierarchy abstract machine framework.

Feudal Learning (Dayan et al., 1993) is a HRL method in which a high-level manager learns to set goals for a set of lower-level workers. The manager provides these goals to the workers, who then use their own local policies to achieve them. The manager receives feedback on the workers' progress and updates its own policy to improve overall performance.

Option Framework (R. S. Sutton et al., 1999) is another HRL method that focuses on learning reusable sub-policies, called options. These options are defined as Markov Decision Processes (MDPs) themselves, with their own reward functions and termination conditions. The main agent can use these options to accomplish its own goals, allowing for a more efficient use of learning time.

MAXQ (Dietterich, 2000) is a HRL approach that decomposes an MDP into a set of subtasks, each of which can be solved independently. The subtasks are organized in a hierarchy, with the highest-level task being the main goal. Each subtask has its own MDP, which can be solved using any RL algorithm. The main agent learns to select which subtask to execute at any given time, taking into account the constraints of the subtask hierarchy.

Finally, Hierarchical Abstract Machine (HAM) approach (Parr et al., 1998) uses a hierarchical structure of finite state machines to represent complex tasks. Each machine represents a subtask and has its own inputs and outputs. At each level of the hierarchy, a decision is made about which subtask to execute, based on the current state of the task.

HRL for dialogue management

HRL can result in faster convergence, better generalization and improved sample efficiency compared to flat RL algorithms (Nachum et al., 2019; Z. Wen et al., 2020). This approach has been adopted in the field of dialogue management research, by applying HAM (Cuayáhuitl, 2009), feudal learning (Casanueva et al., 2018a; Paweł Budzianowski et al., 2017; Casanueva et al., 2018b) or option framework (Peng et al., 2017) and by working on different technical challenges such as transfer learning across different domains (Gasic et al., 2015), domain selection (Cuayáhuitl et al., 2016), or composite task-completion with subgoal discovery (Tang et al., 2018).

Beyond the dialogue management task, various methods of HRL has been proposed with DL as hierarchical DRL (Kulkarni et al., 2016), option-critic architecture (Bacon et al., 2016), stochastic neural networks for HRL (Florensa et al., 2016).

Advanced techniques in RL

RL methods need to strike a balance between exploration and exploitation in order to explore relevant solutions and exploit them quickly. Classical ways to explore environment are epsilon-greedy, Boltzmann and Thompson samplings. More recent methods propose new exploration strategies as bootstrapped DQN (Osband et al., 2016), curiosity-driven exploration (Houthooft et al., 2017; Pathak et al., 2017; Wesselmann et al., 2019), noisy network (Fortunato et al., 2017), bayes by back-propagation neural network (Lipton et al., 2017) and this list can be extended to any bayesian neural network (Jospin et al., 2022).

Unlike deterministic policies that systematically choose the optimal action, deep energy-based policies are proposed methods for learning directly stochastic policies such as Soft Q-learning (Haarnoja et al., 2017) and Soft Actor-Critic (Haarnoja et al., 2018). They assign an energy value to each state-action pair, with lower energy values corresponding to more desirable states or actions.

The list of techniques advanced in RL in the last decade is far too long to be presented exhaustively in this thesis. We invite the curious reader to consult the overview of deep reinforcement learning written by Y. Li (2018).

2.5 Challenges of Reinforcement Learning

Even if RL is a powerful tool for solving complex decision-making problems, it also has several challenges that need to be considered.

The *Deadly Triad*

The concept of *Deadly Triad* was formalised by R. Sutton and Barto (2018) that argue that function approximation, bootstrapping and off-policy learning combined together imply instability and divergence in the learning process. However, it is difficult to give up these practices:

Function approximation, that is *a powerful, scalable way of generalising from a state space much larger than the memory and computational resources*, is necessary for scalability and generalisation.
Bootstrapping, that *updates targets that include existing estimates rather than relying exclusively on actual rewards and complete returns*, is necessary for computational and data efficiency.
Off-policy learning, that is based on *training on a distribution of transitions other than that produced by the target policy*, is necessary for freeing behaviour policy from target policy.

The Deadly Triad can be avoided by eliminating any one of its three elements. However, giving up function approximation sacrifices the ability to handle large

problems and expressive power. Dropping bootstrapping sacrifices computational and data efficiency. Although on-policy methods are sufficient, off-policy learning enables the agent to learn about various policies simultaneously, including policies other than the one it follows. Therefore the challenge is central when designing RL algorithms in order to retain these properties while being robust to instabilities. Especially, recent research focuses on studying and breaking the Deadly Triad with empirical and theoretical support (Hasselt et al., 2018; S. Zhang et al., 2021).

Dealing with the challenges of task-oriented dialogue problems

RL algorithms for dialogue problems have their intrinsic challenges. We distinguish two different approaches to dialogue strategy learning depending on which context the agent gathers its experiences: model-based and simulation-based approaches.

On one hand, model-based learning is done *off-line* without any interaction between the agent and the environment (also called *batch RL*) usually using Maximum Likelihood Estimation (MLE). To do that, it is necessary to obtain a large-scale annotated corpus that is often a difficult task because it requires specific knowledge and expertise and because annotating fine-grained tags for the corpus is a time-consuming and costly process that requires a significant amount of human resources. Thus, one of the main challenges in building task-oriented dialogue systems is to improve data efficiency, especially when resources are limited (Z. Zhang et al., 2020). Similarly, learning from fixed datasets has a number of important shortcomings, as they are not large enough to reliably estimate all transition probabilities, as they are limited to fixed state-action combinations which do not guarantee that the optimal strategy is present in the learning corpus, and as the learning is limited to domains where working systems already exist (Rieser and Lemon, 2011).

On the other hand, simulated-based learning is performed *on-line* by interacting with a simulated environment (also called *model-free* RL) instead of real learning environment in which the agent optimises on-the-fly a policy based on its interactions with human users: in a real learning environment, direct interaction is painful especially when the agent starts to learn a policy, which reinforces the need for improved data efficiency. One way around this problem is to replace it with a simulated user and an error model designed by hand or trained with SL techniques. The main advantages of this approach are that the simulator does not limit the number of training episodes to be generated and that it allows the exploration of strategies that are not included in the training data. However, as the dynamics of the environment are intrinsically determined by the user, designing a simulator that approximates human behaviour is another equally complex task. The reward function $\mathcal{R}$ has to be explicitly constructed and is mostly hand-crafted, tuned and not easily adjustable. Similarly, the transition function $\mathcal{P}$ is also hand-crafted and it is complex, unknown, can vary a lot from one individual to another and it is not clear how it aligns with the user's actual behaviour (Carrara, 2019; Paek et al., 2008b).

2.5.1 Challenges addressed in the scope of this thesis

The ability to learn from few interactions is essential in dialogue applications because human interactions are scarce and costly. Unfortunately, standard RL algorithms usually require a large amount of interactions with the environment to reach good performance. Especially in multi-domain multi-task oriented dialogue scenarios, the ability to adapt a policy from one domain or task to another is relevant in dialogue applications when dialogue policies are similar thus avoiding learning individual policies in favour of a generic policy for all dialogue domains or tasks.

Therefore, dialogue policy learning must meet the needs of **scalability** and **efficiency**. Scalability can be achieved by using data collected from different dialogue tasks and by utilising a generic dialogue policy in a hierarchical way (HRL). Efficiency can be achieved by deploying an advanced off-policy actor critic algorithm combined with efficient exploration techniques based on imitation (IL).

The article written by Le et al. (2018b) introducing *Hierarchical Imitation and Reinforcement Learning* is the first step in our journey. In the case where a hierarchical structure can emerge from the considered problem, the authors show that hierarchical guidance is an effective way to exploit expert feedback to learn sequential decision-making policies, in which the feedback from the high-level expert is used to guide the low-level learner.

This approach is in line with methods that use expert demonstrations to guide the agent in relevant situations to be learned such as DAgger (Stéphane Ross et al., 2011), AggraVate (Stéphane Ross et al., 2014) and LOLS (Chang et al., 2015). To go further in the hybridisation of IL and RL strategies, Hester et al. (2017) propose Deep-Q Learning from Demonstration (DQfD), an extension to DQN that uses expert demonstrations to guide the agent directly in the RL process.

However, noisy demonstrations can disrupt the joint optimization of expert imitation and reward exploitation. Recent methods try to learn a stochastic policy from imperfect demonstrations during the pre-training phase in a video game context (Y. Gao et al., 2019). In multi-domain task-oriented dialogue, Gordon-Hall et al. (2020) leverages the DQfD approach by using experts who are trained with weak demonstrations and then refined during the RL training.

At the same time, the divide-and-conquer strategy enabled by hierarchical decomposition is a catalyst for learning. In the multi-domain task-oriented dialogue, this is a commonly used strategy for managing multiple domains in a single dialogue or managing multiple slots in a single domain.

However, this strategy, which involves independence between domains and between slots, does not benefit from knowledge transfer. More recently, a series of work on single-domain task-oriented dialogues has demonstrated that using a graph structure instead of a tree structure (to be understood as feudal learning) can accelerate the learning process in RL settings (L. Chen et al., 2018; Z. Chen et al., 2020b; Z. Chen et al., 2020a).

Here, we are interested on how structure the architecture and the learning of dialogue management models in order to reach faster good performance not only in single-domain-task but in **multi-domain-task environments**. This is done by means of a strategy combining IL and RL and by means of a Graph Neural Network (GNN) architecture.

II

Hierarchical Imitation and Reinforcement Learning for Task-Oriented Dialogue Management

3

Guiding Dialogue Policies with Diluted Expert Demonstrations

A learning dialogue agent can infer its behaviour from interactions with the users. These interactions can be taken from either human-to-human or human-machine conversations. However, human interactions are scarce and costly, making learning from few interactions essential. One solution to speedup the learning process is to guide the agent's exploration with the help of an expert. It can benefit from demonstrations in a supervised fashion to further improve exploration. However, some common practices are undesirable if we want to learn in real-context with human interactions and noise. We present in this chapter several hybrid imitation and reinforcement learning strategies for dialogue policy where the guiding expert is a near-optimal handcrafted policy. We incorporate these strategies with state-of-the-art reinforcement learning methods based on Q-learning and actor-critic. We notably propose a randomised exploration policy which allows for a seamless hybridisation of the learned policy and the expert, which can be seen as a dilution of the expert's demonstration into the resulting policy. Our experiments show that our hybridisation strategy outperforms several baselines, and that it could accelerate the learning when facing real humans.

3.1 Motivation

The ability to learn from few interactions is essential in dialogue applications because human interactions are scarce and costly. Unfortunately, standard RL algorithms usually require a large amount of interactions with the environment to reach good performances. One solution to speedup the learning process is to guide the agent's exploration. We also claim that policy learning should not be deterministic. By taking a closer look at the way dialogue policy learning works, we observed that some actions do not contribute to the dialogue and therefore do not change the current state. We hence posit that greedy off-policy learning methods suffer this situation and need a randomised policy rather than a deterministic one. This also can be

profitable for a better exploration strategy than epsilon-greedy. Moreover, former works did used stochastic policy (e.g. Young et al., 2013a) and recent methods show that stochastic learning policies like soft-kind policies can perform very well and present elegant proprieties (Haarnoja et al., 2017; Haarnoja et al., 2018; Y. Gao et al., 2019). The first question we address here is: can dialogue policy learning be improved with stochastic off-policy learning methods?

Secondly, we think that external demonstrations should not be reserved only for supervision and instead should be directly integrated in the RL process. Several methods search a way to exploit demonstrations to accelerate the learning with the conviction that demonstrations are the solution to the sparse reward. These methods are presented as useful and efficient as showed in Hester et al., 2017. They can be based on Imitation Learning (IL) to learn an optimal policy function or on Inverse Reinforcement Learning (IRL) to learn an optimal reward function. Nevertheless, the majority of these approaches uses Supervised Learning (SL) and compares them with pure reinforcement methods, which is not ideal because: (i) the integrity of policy learning is safe obviously thanks to the refinement of the policy in the RL environment; (ii) a supervised update in policy learning is more powerful than a reinforced one (Hester et al., 2017; Su et al., 2016a; Y. Gao et al., 2019). In addition, SL strongly constrains the learning of the agent according to the demonstrations and the performance of these methods therefore depends on the quality of the demonstrations.

Although warm SL can obviously improve the performance, demonstrations may still be useful to guide efficiently the exploration even when they are sub-optimal. Similarly, we think that exploration with demonstrations can improve learning. In addition, we consider that human expertise can be used in different ways. The most classical one is to use demonstrations that humans have already produced. Another way is to use a hand-crafted agent that has been designed, evaluated and fine-tuned by humans. Indeed, Casanueva et al., 2017 have shown that hand-crafted approaches still perform better than policy learning approaches. Moreover it is a fact that human interactions and manually crafted rules are not only costly but also time consuming, while simulated interactions are cheaper and easier to collect (Su et al., 2016b; Schatzmann et al., 2006). Therefore, the second issue that raised in our work is: can we use demonstrations without supervision and only as a way to guide exploration in an on-line learning?

Our contribution in this chapter focuses on exploration strategies in pure RL. To the best of our knowledge, this might be the first proposal to learn a dialogue policy with demonstrations given by a sub-optimal expert directly in the learning process without any supervision. We openly propose to learn a randomised exploration policy in order to integrate seamlessly the expert actions and to avoid being stuck on a deterministic policy. The process expert's demonstrations can be viewed as diluted in the derived policy (in contrast to bootstrap it).

3.2 Hybrid imitation and reinforcement learning for dialogue policies

The proposed approach to policy dialogue learning is based on recent DL methods to which we bring the advantages of stochastic techniques, namely Boltzmann sampling, and to which we integrate demonstrations directly into the RL process in order to better guide exploration and make relevant exploitation. RL is usually implemented as either value-based method as Q-learning or policy-based method as actor-critic. Both are our baselines with which we will make our contributions. During the RL process, demonstrations can serve as an efficient way to explore the environment even if not optimal. Indeed, they can lead the agent to receive rewards promptly and so can lead it to exploit confident winning trajectories quickly.

We propose different demonstration sampling strategies corresponding to different ways of using the knowledge of an adviser expert. They are drawn on IL techniques but they are not designed in a supervised fashion (except for the last one). These hybrid IL and RL methods are distinguished by (i) their ability to exploit human data as a teacher, (ii) their ability to outperform their teacher on the long run, and (iii) their convergence speed. In the following, we will examine how the loss function $L(\Theta;\pi,\pi_E)$ is computed with respect to the agent policy π, the oracle policy π_E and the demonstration sampling strategy f in order to optimise the parameters of the model Θ:

$$\begin{aligned} L(\Theta;\tilde{\pi}) &= \lambda_{RL} L_{RL}(\Theta;\tilde{\pi}) + \lambda_{IL} L_{IL}(\Theta;\tilde{\pi}) \\ \text{with} \quad \tilde{\pi} &\sim f(\pi,\pi_E) \end{aligned} \tag{3.1}$$

3.2.1 Pure reinforcement learning

The first approach is a *Pure Reinforcement Learning* method which improves the agent policy according to its own trajectories in order to accumulate more rewards in the future and thus does not depend on the oracle policy as detailed in Algorithm 3.1 and shown in Figure 3.1. Therefore, the loss function is defined by the Equation 3.2:

$$\text{Eq. (3.1)} \quad \text{with} \quad \begin{cases} \lambda_{RL} = 1, \\ \lambda_{IL} = 0, \\ \tilde{\pi} \sim \pi \end{cases}$$

$$L(\Theta;\tilde{\pi}) = L_{RL}(\Theta;\pi) \tag{3.2}$$

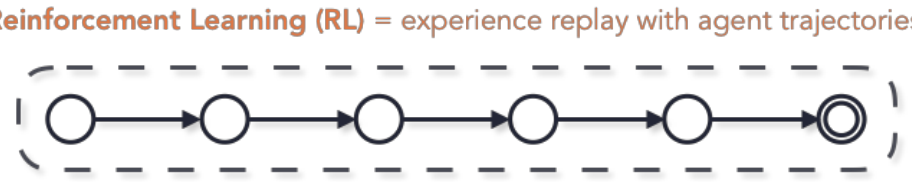


Figure 3.1: RL is based on the experience replay from the agent's trajectories. The states are represented by circles (double circle for the terminal state) and the agent's actions by black arrows.

Algorithm 3.1: Pure Reinforcement Learning

Input: $\mathcal{D} \leftarrow \emptyset$, an empty dataset
$(\pi, Q) \leftarrow$ any policy and value functions parametrised by Θ
$L \leftarrow$ any loss function to optimise with respect to $\mathcal{D}$ and Θ
Output: trained policy π with pure reinforcement learning
for each train step **do**
for each episode i **do** ▷ Sample T-step trajectories using π
$\tau_i \leftarrow \{(s_t, a_t, r_{t+1}, s_{t+1})\}_{0 \leq t \leq T}$
end for
$\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}_{1 \leq i \leq n}$ ▷ Update the train dataset $\mathcal{D}$
$\pi, Q \leftarrow \textsc{Optimise}(L, \mathcal{D}, \Theta)$ ▷ Train the policy π on $\mathcal{D}$
end for

The Q-learning baseline is the combination of Double and Duelling Deep-Q-Network with Experience Replay (DQN/D3QN) (Hasselt et al., 2015; Ziyu Wang et al., 2016). The Double DQN architecture is an alternative DQN method to mitigate the problem of overoptimistic value estimation and the Duelling DQN architecture is for learning more efficiently by decoupling value and advantage functions. On top of that experience replay can be added which is a popular technique for reducing sample correlation and for improving sample efficiency.

The actor-critic baseline is the Trust Region Policy Optimization for Actor-Critic with Experience Replay (A2C/ACER/TRACER) (Ziyu Wang et al., 2017; Weisz et al., 2018). This method is based on the *Policy Gradient Descent* (R. Sutton et al., 2000) for which we bring computation improvements. The importance sampling truncation with bias correction technique is used to correct the perceived sampling distribution induced by experience replay in order to reduce variance. We apply the Retrace algorithm (Munos et al., 2016) to recursively estimate the advantage function in safe and efficient way with small bias and variance. Finally, we use the Trust Region Policy Optimisation (TRPO) (Schulman et al., 2017a) for adjusting the policy gradient in order to learn in a safe parameters region limiting the deterioration of the policy performance.

3.2.2 Behaviour cloning for imitation learning

Behaviour cloning (BC) is a pure supervised learning method that tries to mimic the oracle policy as shown in Algorithm 3.2. Its loss function is the cross-entropy loss function as in a classification problem (see Equation 3.3). A BC agent can be trained from static human or simulated demonstrations as illustrated in Figure 3.2.

$$\text{Eq. (3.1)} \quad \text{with} \quad \begin{cases} \lambda_{RL} = 0, \\ \lambda_{IL} = 1, \\ \tilde{\pi} \sim \pi_E \end{cases}$$

$$L(\Theta; \tilde{\pi}) = L_{IL}(\Theta; \pi_E) \tag{3.3}$$

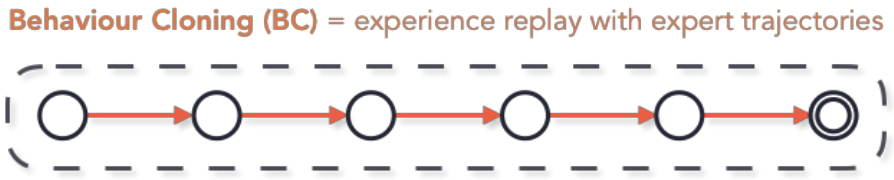


Figure 3.2: Behaviour cloning is based on the experience replay from the oracle's trajectories. The states are represented by circles (double circle for the terminal state) and the oracle's actions by orange arrows.

Algorithm 3.2: Behaviour Cloning for Imitation Learning

Input: $\mathcal{D} \leftarrow \emptyset$, an empty dataset
$(\pi, Q) \leftarrow$ any policy and value functions parametrised by Θ
$L \leftarrow$ any loss function to optimise with respect to $\mathcal{D}$ and Θ
Output: trained policy π with behaviour cloning
for each train step **do**
  for each episode i **do** ▷ Sample T-step trajectories using π_E
    $\tau_i \leftarrow \{(s_t, a_t)\}_{0 \leq t \leq T}$ ▷ Keep only state-action transition
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}_{1 \leq i \leq n}$ ▷ Update the train dataset $\mathcal{D}$
  $\pi, Q \leftarrow \text{OPTIMISE}(L, \mathcal{D}, \Theta)$ ▷ Train the policy π on $\mathcal{D}$
end for

BC benefits from a more efficient learning capacity than RL since the initial state-action evaluation problem becomes a state-optimal action classification problem. But this advantage is limited insofar as the oracle delivers optimal demonstrations since, by construction, it cannot outperform its teacher.

3.2.3 Imitation learning from oracle demonstrations

Imitation Learning from Oracle Demonstrations (ILfOD) is a reinforcement learning method with **online data augmentation** which allows the agent to play oracle actions as demonstrations and inject them in its replay buffer (Figure 3.3). This is why the loss function is always the same as RL except that the policy followed is a mixture between the agent and the oracle (Equation 3.4). In practice, before each new episode, the policy to be followed is chosen among those of the agent or the expert to generate the next trajectory (Algorithm 3.3).

$$\text{Eq. (3.1)} \quad \text{with} \quad \begin{cases} \lambda_{RL} = 1, \\ \lambda_{IL} = 0, \\ \tilde{\pi} \sim \beta\,\pi + (1-\beta)\,\pi_E \quad \text{sampling at the start of each dialogue} \end{cases}$$

$$L(\Theta; \tilde{\pi}) = L_{RL}(\Theta; \beta\,\pi + (1-\beta)\,\pi_E) \tag{3.4}$$

Figure 3.3: ILfOD is based on the experience replay composed of both oracle and agent trajectories (represented by two different datasets). The states are represented by circles (double circle for the terminal state), the oracle's actions by orange arrows and the agent's actions by black arrows.

Algorithm 3.3: Imitation Learning from Oracle Demonstrations

Input: $\mathcal{D} \leftarrow \emptyset$, an empty dataset
$(\pi, Q) \leftarrow$ any policy and value functions parametrised by Θ
$L \leftarrow$ any loss function to optimise with respect to $\mathcal{D}$ and Θ
Output: trained policy π with ILfOD
for each train step **do**
  for each episode i **do** ▷ Sample T-step trajectories using π or π_E
    $\pi_i \leftarrow \textsc{Sample}(\pi, \pi_E)$ ▷ Draw the policy to be played
    $\tau_i \leftarrow \{(s_t, a_t, r_{t+1}, s_{t+1})\}_{0 \leq t \leq T}$ ▷ Sample one trajectory with π_i
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}_{1 \leq i \leq n}$ ▷ Update the train dataset $\mathcal{D}$
  $\pi, Q \leftarrow \textsc{Optimise}(L, \mathcal{D}, \Theta)$ ▷ Train the policy π on $\mathcal{D}$
end for

Both BC and ILFOD can be trained from static human demonstrations. However, this time, ILFOD can outperform its teacher through policy refinement in the reinforcement learning environment. This technique therefore allows us to provide the agent with promising demonstrations to quickly obtain rewards. In other point of view, the agent learns as if they are two datasets. The first one contains its trajectories and the second the expert demonstrations.

In our experiments, we share in the buffer the agent's own actions as well as those generated by the oracle. We propose that in β (in percent) of dialogues, the agent plays for itself. Otherwise in $1-\beta$ of dialogues, the expert gives to the agent one of its expert trajectories as demonstration and the agent replays the dialogue as if it was the one who played it.

3.2.4 Imitation learning from oracle feed-backs

Imitation Learning from Oracle Feed-backs (ILfOF) is also a reinforcement learning method with **online data augmentation**. However, unlike ILfOD, the agent is allowed to be guided by oracle actions at any dialogue turn and inject them into its replay buffer. We draw on the exploration strategy of the DAgger method (Stéphane Ross et al., 2011) but apply it in an RL process with mixed initiatives between the oracle and the agent during a dialogue (see these differences in Equation 3.5 and Algorithm 3.4). In other words, the agent plays in an RL environment and the oracle can interrupt the agent to give feedback, i.e. its optimal action (Figure 3.4).

$$\text{Eq. (3.1)} \quad \text{with} \quad \begin{cases} \lambda_{RL} = 1, \\ \lambda_{IL} = 0, \\ \tilde{\pi} \sim \beta\,\pi + (1-\beta)\,\pi_E \quad \text{sampling at the start of each dialogue turn} \end{cases}$$

$$L(\Theta;\tilde{\pi}) = L_{RL}(\Theta;\beta\,\pi + (1-\beta)\,\pi_E) \tag{3.5}$$

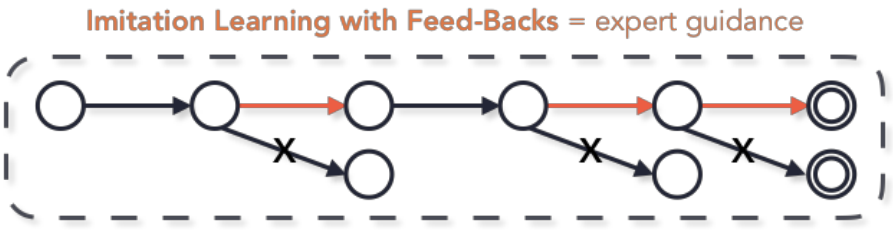


Figure 3.4: ILfOF is based on the experience replay from the hybrid trajectories between the oracle and the agent. The states are represented by circles (double circle for the terminal state), the oracle's actions by orange arrows and the agent's actions by black arrows. Arrows with a cross indicate that the agent has been guided by the oracle.

Algorithm 3.4: Imitation Learning from Oracle Feed-backs

Input: $\mathcal{D} \leftarrow \emptyset$, an empty dataset
$(\pi, Q) \leftarrow$ any policy and value functions parametrised by Θ
$L \leftarrow$ any loss function to optimise with respect to $\mathcal{D}$ and Θ
Output: trained policy π with ILfOF
for each train step **do**
  for each episode i **do** ▷ Sample T-step trajectories using π or π_E
    for each turn t **do**
      $\pi_t \leftarrow \textsc{Sample}(\pi, \pi_E)$ ▷ Draw the policy to be played
      $(s_t, a_t, r_{t+1}, s_{t+1}) \sim \pi_t$ ▷ Sample one turn with π_t
    end for
    $\tau_i \leftarrow \{(s_t, a_t, r_{t+1}, s_{t+1})\}_{0 \leq t \leq T}$ ▷ Collect the mixed trajectory
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}_{1 \leq i \leq n}$ ▷ Update the train dataset $\mathcal{D}$
  $\pi, Q \leftarrow \textsc{Optimise}(L, \mathcal{D}, \Theta)$ ▷ Train the policy π on $\mathcal{D}$
end for

This technique therefore allows us to provide the agent with promising dialogue actions to quickly explore the environment by been redirected to more relevant oracle action at any time, as DAgger does. As we do with ILfOD, we let the agent and the oracle share the initiative. We propose that in β (in percent) of dialogue actions, the agent plays for itself. Otherwise in $1-\beta$ of dialogue actions, the expert gives to the agent its expert action as feed-back and the agent plays the dialogue action as if it was its choice.

3.2.5 Imitation learning from oracle supervision

Imitation Learning from Oracle Supervision (ILfOS) is a **combination of supervised and reinforced learning** based on ILfOD for data augmentation (Equation 3.6) and inspired by *Deep Q-learning form Demonstrations* (DQfD) (Hester et al., 2018) in which the reinforced loss function is augmented with a supervised loss called *margin loss* (Equation 3.7). This modification allows the agent to benefit from demonstration-guided exploration and teacher-forced learning if its best action is not sufficiently aligned with that of the oracle (Figure 3.5).

$$\text{Eq. (3.1)} \quad \text{with} \quad \begin{cases} \lambda_{RL} = 1, \\ \lambda_{IL} = 1, \\ \tilde{\pi} \sim \beta\,\pi + (1-\beta)\,\pi_E & \text{sampling at the start of each dialogue} \end{cases}$$

$$L(\Theta; \tilde{\pi}) = L_{RL}(\Theta; \beta\,\pi + (1-\beta)\,\pi_E) + L_{IL}(\Theta; \beta\,\pi + (1-\beta)\,\pi_E) \tag{3.6}$$

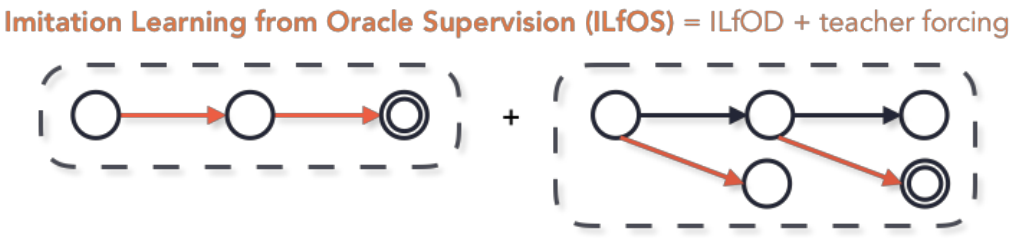


Figure 3.5: ILfOS is based on the experience replay from the oracle and agent's trajectories as ILfOD but with teacher forcing learning. The states are represented by circles (double circle for the terminal state), the oracle's actions by orange arrows and the agent's actions by black arrows.

Algorithm 3.5: Imitation Learning from Oracle Supervision

Input: $\mathcal{D} \leftarrow \emptyset$, an empty dataset
$(\pi, Q) \leftarrow$ any policy and value functions parametrised by Θ
$L \leftarrow$ any loss function to optimise with respect to $\mathcal{D}$ and Θ
Output: trained policy π with ILfOS
for each train step **do**
for each episode i **do** ▷ Sample T-step trajectories using π or π_E
$\pi_i \leftarrow \text{SAMPLE}(\pi, \pi_E)$ ▷ Draw the policy to be played
$\tau_i \leftarrow \{(s_t, a_t, a_t^*, r_{t+1}, s_{t+1})\}_{0 \leq t \leq T}$ ▷ Sample one trajectory with π_i and keep the optimal action
end for
$\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}_{1 \leq i \leq n}$ ▷ Update the train dataset $\mathcal{D}$
$\pi, Q \leftarrow \text{OPTIMISE}(L, \mathcal{D}, \Theta)$ ▷ Train the policy π on $\mathcal{D}$
end for

As DQfD approach (Hester et al., 2018), ILfOS agent learns with a supervised loss named *margin loss* (Equation 3.7) which forces to associate to the oracle action a better score by artificially applying a margin:

$$L_{IL}(\Theta, \pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{(s_t, a_t^*) \in \tau} \max_{a_t \in \mathcal{A}}[Q_\Theta(s_t, a_t) + l(a, a_t^*)] - Q_\Theta(s_t, a_t^*)\right] \quad (3.7)$$
$$\text{with} \quad l(a_t, a_t^*) = \mu * \mathbf{1}_{a_t \neq a_t^*}$$

with Q_Θ is any score function (value function or log-policy function), s_t the current state, a_t and a_t^* the current agent and oracle actions and μ the margin penalty.

An ILfOS agent converges usually faster than an ILfOD agent, but because they require a constant or frequent access to the oracle's decisions, they cannot exploit static human logs (as shown in Algorithm 3.5).

3.3 Stochastic Exploration Strategy

We present our contributions related to the exploration strategy, which we integrate to the hybrid imitation and reinforcement off-policy learning methods explained earlier. Indeed, exploring with a smooth stochastic policy seems to be more adapted than exploring with a deterministic one or with ϵ-greedy for instance. We propose to learn an energy-based policy in the dialogue environment where the agent samples his actions according to Boltzmann's stochastic sampling.

Energy-based policies modelling: Inspired from Haarnoja et al. (2017), we opt for using energy-based policies of the following form:

$$\pi(a_t|s_t) \propto \exp(-\mathcal{E}(s_t, a_t)) \tag{3.8}$$

where $\mathcal{E}$ is the energy-based function that can be related to any score function. When using value-based method, the energy function can be represented by the Q-function with parameter τ, called the temperature, where we set $\mathcal{E}(b_t,a_t) = -\frac{1}{\tau}Q^{\pi}(b_t,a_t)$. When using policy-based method, the energy function is directly represented by the log-policy and so is learned implicitly.

In theory, learning an energy-based policy function is motivated by the consideration of the maximum entropy RL setup in which the rewards are augmented with an entropy term (Haarnoja et al., 2017). Even if there exists an interesting relationship between the temperature and the relative importance of entropy over rewards, we decide to separate the maximum entropy objective from maximum rewards objective which leads to maximizing entropy without discount factor. In other words, we maximize the entropy at all time step without taking into account the time decay and so we promote exploration at any time step and not only at the beginning.

Energy-based sampling: A stochastic sampling with energy-based policies can be achieved by the Boltzmann sampling. Contrary to the commonly used strategy as ϵ-greedy where actions a_t are sampled as follows:

$$a_t \sim (1-\epsilon) \arg\max_{a \in \mathcal{A}} \pi(a|s_t) + \epsilon U(\mathcal{A}) \tag{3.9}$$

where $U(\mathcal{A})$ is the uniform distribution over action space, the Boltzmann sampling strategy samples actions from:

$$a_t \sim \exp(-\mathcal{E}(s_t,\cdot)) = \pi(\cdot|s_t) \tag{3.10}$$

We think that exploring with smooth stochastic policy should improve the stability of learning in spite of exploitation strength. Indeed, stochastic sampling is equivalent to play current policy when greedy sampling is equivalent to play hypothetical optimal policy with respect to the current policy.

One of its advantages is that policy learning is less influenced by the policy function changes. Conversely, the ϵ-greedy sampling faces sudden jumps in action choices due to the argmax operator (see Figure 3.6). So in theory, the Boltzmann strategy can make learning more stable than the ϵ-greedy strategy.

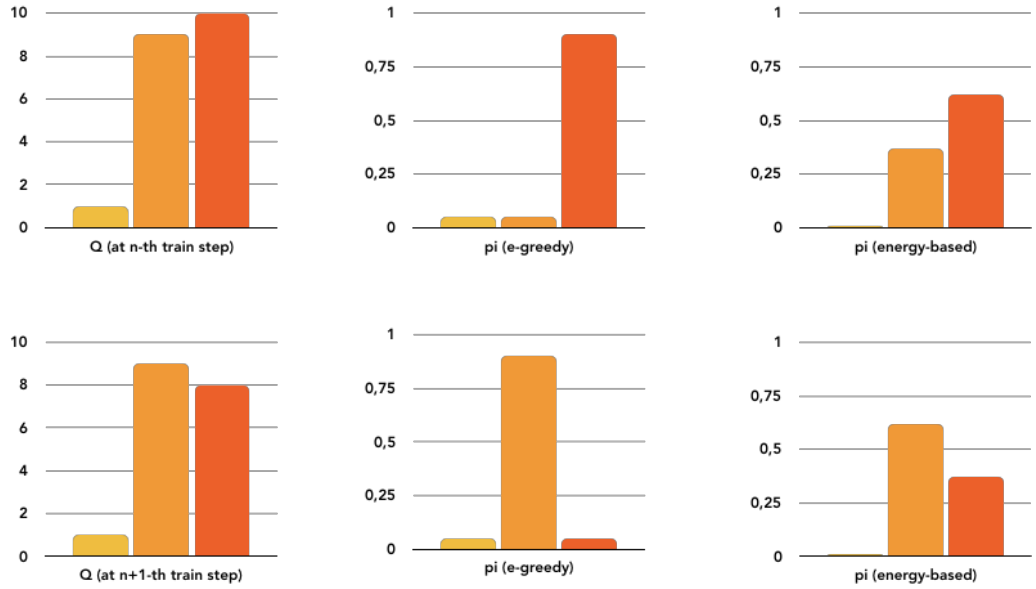


Figure 3.6: Policy learning is less influenced by the policy function changes (from the first row to the second). After the score function changes, the ϵ-greedy sampling faces sudden jumps when the Boltzmann sampling is more stable.

Another advantage is that the temperature parameter can control the exploration-exploitation balance. So, to counter the weakness of its exploitation, it can be interesting to define correctly the temperature.

In practice, we decide to add random exploration, namely ϵ-Boltzmann sampling, in such a way that the actions are sampled according to:

$$a_t \sim (1-\epsilon)\pi(\cdot|s_t) + \epsilon U(\mathcal{A}) \tag{3.11}$$

with decreasing ϵ parameter in order to explore enough before following the stochastic policy with a fixed τ temperature parameter.

In all cases, the sampling of our agent's actions is combined with the sampling of the oracle's actions according to the hybrid strategy presented before as follows:

$$\tilde{\pi} \sim \beta\left[(1-\epsilon)\pi + \epsilon\,\mathcal{U})\right] + (1-\beta)\pi_E \tag{3.12}$$

In particular, a hand-crafted agent is used to simulate near-optimal demonstrations and feed-backs. This one is not optimal as seen in the benchmark of PyDial (Casanueva et al., 2017) but it offers good performance compared with the other deep learning methods. Also, it has been designed, evaluated and fine-tuned by humans. So it is a way to mimic human expertise and can be served as a near-optimal expert in our experimentation.

3.4 Experimentation

We propose to organise our study along two axes that will frame our experiments: are the demonstrations and feedbacks sufficient for the exploration in the reinforcement learning paradigm and how good are demonstrations instead of supervision? Indeed, on the one hand, ILfOD and ILfOF are close from a statistical point of view but differ on the moment of interrogating the oracle to guide the agent. We thus propose in Subsection 3.4.1 to study this problem in single-domain task-oriented dialogues in order to determine which one is the best in place of pure RL.

Moreover, imitation is useful when reinforcement is not sufficient. With this in mind, we propose in Subsection 3.4.2 to examine this issue in multi-domain multi-task-oriented dialogues to determine whether supervision is the only way to solve a more difficult dialogue problem or whether demonstrations are sufficient as a catalyst to solve this problem.

3.4.1 Are the demonstrations sufficient for the exploration?

In our experiments, the PyDial framework (Ultes et al., 2017b) is used, which implements an agenda-based user simulation (Schatzmann et al., 2007). As in Casanueva et al. (2017) we tested our algorithms for policy learning on different domains and in different environments by increasing the inputs' noise. As shown in previous Chapter 1, the domains differ from each other by the ontology size, impacting the state and action space dimensions.

In particular, we decide to evaluate our policy models according to three levels of noise with respect to the semantic error rate (SER). This corresponds to the noise that comes from the ASR and the NLU channels. In PyDial, this is modelled at the semantic level whereby the true user action is corrupted by noise to generate an N-best-list with associated confidence scores. This corresponds to the environments 1, 3 and 6 proposed by Casanueva et al. (2017)

Methods: Table 3.1 shows the compared policy models. HDC corresponds to the handcrafted policy learning, which is a frame-based approach written by experts. DQN and ACER are the baselines enhanced with stochastic (Stoc) exploration and either demonstrations (ILfOD) or feed-backs (ILfOF).

Hyper-parameters and Training: All algorithms use by default ϵ-Boltzmann sampling strategy during training and testing. We use common parameters between the different algorithms as learning rate $\alpha = 1 \cdot 10^{-4}$, experience replay buffer of size 10 000, train batch of size 128, discount factor $\gamma = 0.9$, linearly decreasing epsilon $\epsilon \in [0.05; 0.55]$, fixed temperature $\tau = 100$, weights dropout with probability 0.1, oracle-agent play ratio $\beta = 0.5$. Others parameters related to referred papers are chosen identically. All deep networks have the same architecture composed by two

Method name	Abbrev.
Handcrafted Policy	HDC
Stochastic Q-learning	Stoc-DQN
Stochastic Q-learning with Demonstrations	Stoc-DQN-ILfOD
Stochastic Q-learning with Feed-backs	Stoc-DQN-ILfOF
Stochastic Actor-Critic	Stoc-ACER
Stochastic Actor-Critic with Demonstrations	Stoc-ACER-ILfOD
Stochastic Actor-Critic with Feed-backs	Stoc-ACER-ILfOF

Table 3.1: Overview of proposed methods for single-domain task-oriented dialogues.

hidden layers with respectively 128 and 64 neurons. The Adam optimiser was used to train all the deep-RL models. The first hidden layer is shared between learning modules depending on the algorithms. Each network is trained after one dialogue with one training step.

As in Casanueva et al. (2017) the maximum dialogue length was set to 25 turns and the discount factor γ was 0.9. We evaluated our policy models on the the average success rate and average reward. Success rate is defined as the percentage of dialogues which are completed successfully, i.e. whether the DM is able to fulfill the user goal or not. More details on the reward function used (Equation 2.6) and evaluation measures can be found in the Chapter 1 and Chapter 2.

Experiments: First, we performed a short training stage over 1 000 dialogues during which the agent learns from its interactions with the environment and potentially from demonstrations or feed-backs given by the simulated expert. We compare the performances with the Hard-DQN which uses a deterministic sampling and with eNAC (episodic Natural Actor-Critic) which uses the true natural policy gradient. Both are given by Casanueva et al. (2017). At this stage we search to answer the question: can we improve dialogue policy learning with demonstrations directly in the reinforcement learning process?

Second we performed a long training stage over 10 000 dialogues. This will evaluate the contribution of stochastic sampling strategy during training and testing stages. Here we search to answer the question: can we improve dialogue policy learning with stochastic off-policy learning methods in order to compete the handcrafted agent?

All methods are evaluated after training over 1 000 dialogues during which the learned policy is fixed. For the second experiments, we decide to compute the average performance over the last five checkpoints from dialogue indices 6 000 to 10 000 with a step of 1 000 dialogues. This calculation is done in order to reduce variance induced by RL when we estimate the performance of the models.

Results: The results of the first experiments are presented in Table 3.2. We can observe that demonstrations can improve the performance during the early training when compared with baselines (i.e eNAC and Hard-DQN). Particularly, our proposed stochastic ACER models outperform eNAC baseline on all environments. Although stochastic ACER is already efficient to learn in early training without demonstrations, we can notice that the best performances are obtained through an expert-guided policy.

		Hard-DQN		Stoc-DQN		Stoc-DQN-ILfOD		Stoc-DQN-ILfOF		HDC	
Task		Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.
0% SER	CR	88.6%	11.60	72.7%	8.08	93.4%	12.30	**98.1%**	**13.23**	100.0%	14.00
	SFR	48.0%	2.70	64.5%	3.51	**97.2%**	**11.56**	93.5%	10.69	98.2%	12.40
	LAP	61.9%	5.50	57.6%	2.54	**95.3%**	**11.38**	81.7%	8.41	97.0%	11.70
15% SER	CR	79.5%	9.20	68.4%	5.47	88.8%	10.17	**92.2%**	**10.59**	96.7%	11.00
	SFR	42.4%	1.00	56.1%	-0.79	**93.5%**	**9.34**	87.8%	7.41	90.9%	9.00
	LAP	51.9%	3.10	43.7%	-5.82	87.6%	8.06	**88.0%**	**8.31**	89.6%	8.70
30% SER	CR	72.3%	6.90	64.9%	2.50	82.8%	**8.17**	**83.0%**	7.67	89.6%	9.30
	SFR	35.6%	-1.20	62.3%	1.10	73.5%	3.01	**78.8%**	**4.18**	79.0%	6.00
	LAP	47.5%	1.40	71.0%	1.78	76.1%	4.20	**77.1%**	**4.45**	76.1%	5.30

		eNAC		Stoc-ACER		Stoc-ACER-ILfOD		Stoc-ACER-ILfOF		HDC	
Task		Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.
0% SER	CR	93.0%	12.20	98.5%	**13.47**	**99.1%**	13.40	99.0%	13.46	100.0%	14.00
	SFR	85.8%	9.90	97.4%	**11.77**	**97.9%**	11.71	96.3%	11.50	98.2%	12.40
	LAP	84.2%	8.80	96.5%	11.65	**98.7%**	12.00	98.4%	**12.11**	97.0%	11.70
15% SER	CR	85.7%	10.00	93.9%	11.17	**95.9%**	**11.72**	94.8%	11.38	96.7%	11.00
	SFR	73.6%	6.20	90.7%	8.88	**92.8%**	**9.11**	90.5%	8.84	90.9%	9.00
	LAP	71.0%	5.50	**93.2%**	**9.60**	83.8%	7.51	89.7%	9.01	89.6%	8.70
30% SER	CR	73.6%	6.70	86.6%	8.82	81.6%	8.15	**89.4%**	**9.26**	89.6%	9.30
	SFR	55.2%	1.40	79.6%	4.62	81.3%	4.92	**83.1%**	**5.46**	79.0%	6.00
	LAP	56.3%	1.90	**81.6%**	**5.79**	78.5%	4.67	79.1%	5.29	76.1%	5.30

Table 3.2: Results of Experiment 1. Short term learning after 1 000 training dialogues for 1 000 testing dialogue. Each bold result represent the best model according to the referenced baseline.

Moreover, the proposed Stoc-DQN policy models outperform also Hard-DQN for all environments. We can also notice that demonstrations clearly improve performance of the models. Indeed, we see that demonstrations efficiently guide the exploration of the models. For instance, Stoc-DQN-ILfOD model outperforms all the others for SFR and laptops in the environment (0% SER) and for SFR in environment (15% SER). Similarly, Stoc-DQN-ILfOF outperforms all the other models for all domains in the environment (30% SER). This attests a faster exploitation of the environment that therefore improves outcomes in more difficult environments.

Finally, we can observe that in some cases the stochastic policy models outperform the handcrafted expert, such as: Stoc-ACER with 15% SER and 30% SER for laptops; Stoc-ACER-ILfOD with 0% SER for laptops and with 15% SER for SFR; Stoc-ACER-ILfOF with 30% SER applied SFR. It is worth noting that laptops contains more constrains than the other domains (Table 1.4). In addition, with 0% and

15% SER the the performance of Stoc-ACER-ILfOD is very close to the expert for CR. This suggest that integrating demonstrations directly in the RL process without supervision can improve dialogue policy learning.

The results of our second experiments are presented in Table 3.3. In most environments, methods learn very well compared to the benchmarks Casanueva et al. (2017). Furthermore, some of them can compete with the handcrafted agent whether it is a stochastic Q-learning approach or a stochastic actor-critic approach. For instance, Stoc-DQN-ILfOD outperforms handcrafted expert with 30% SER for laptops and SFR. Again stochastic ACER was more robust to the different environments, showing better performance, particularly for Stoc-ACER-ILfOD with 30% SER for laptops.

Task		Stoc-DQN Suc.	Stoc-DQN Rew.	Stoc-DQN-ILfOD Suc.	Stoc-DQN-ILfOD Rew.	Stoc-DQN-ILfOF Suc.	Stoc-DQN-ILfOF Rew.	HDC Suc.	HDC Rew.
0% SER	CR	97.48%	12.67	98.34%	13.30	98.88%	13.46	**100.0%**	**14.00**
	SFR	94.18%	10.99	87.40%	9.58	95.62%	10.94	**98.2%**	**12.40**
	LAP	95.08%	11.10	**98.10%**	**11.76**	**98.40%**	**11.77**	97.0%	11.70
15% SER	CR	92.22%	10.29	94.16%	11.26	95.76%	**11.64**	**96.7%**	11.00
	SFR	90.64%	8.56	88.70%	8.06	89.22%	8.21	**90.9%**	**9.00**
	LAP	**90.34%**	8.59	**92.56%**	**9.40**	**91.86%**	**9.19**	89.6%	8.70
30% SER	CR	84.32%	7.73	85.36%	8.64	85.46%	8.65	**89.6%**	**9.30**
	SFR	**82.48%**	5.11	**80.26%**	5.05	**80.34%**	4.63	79.0%	**6.00**
	LAP	**82.24%**	**5.89**	**84.62%**	**6.25**	**83.40%**	**6.00**	76.1%	5.30

Task		Stoc-ACER Suc.	Stoc-ACER Rew.	Stoc-ACER-ILfOD Suc.	Stoc-ACER-ILfOD Rew.	Stoc-ACER-ILfOF Suc.	Stoc-ACER-ILfOF Rew.	HDC Suc.	HDC Rew.
0% SER	CR	99.60%	**14.02**	99.64%	**14.03**	99.30%	13.87	**100.0%**	14.00
	SFR	97.66%	12.36	96.48%	11.98	96.34%	11.98	**98.2%**	**12.40**
	LAP	95.24%	11.23	95.34%	11.28	95.40%	11.24	**97.0%**	**11.70**
15% SER	CR	**97.56%**	**12.69**	95.80%	**12.32**	**96.78%**	**12.59**	96.7%	11.00
	SFR	88.62%	**9.26**	87.64%	8.91	86.98%	8.67	**90.9%**	9.00
	LAP	88.74%	**8.72**	88.00%	8.55	86.26%	8.24	**89.6%**	8.70
30% SER	CR	**89.82%**	**10.19**	89.38%	**9.98**	89.32%	**10.04**	89.6%	9.30
	SFR	72.74%	4.38	78.22%	5.45	71.02%	4.37	**79.0%**	**6.00**
	LAP	75.80%	4.86	**78.66%**	**5.40**	**77.82%**	5.23	76.1%	5.30

Table 3.3: Results of Experiment 2. Long term learning, average from 6 000 to 10 000 training dialogues, for 1 000 testing dialogue. Each bold result represent better models than the handcrafted agent.

These results are encouraging and suggest that stochastic sampling makes it possible to learn a policy that performs as well as the handcrafted agent, which has been designed and fine-tuned by humans, even in hard environments. Also, these results show that demonstrations can significantly contribute to improve the performance at early learning stages.

3.4.2 How good are demonstrations instead of supervision?

In this section, the ConvLab2 framework (Zhu et al., 2020) is used for multi-domain multi-task oriented dialogues to evaluate our approach against delivered baselines. We present the tested methods, the experimental setup and the evaluation metrics.

Methods: Table 3.4 shows the compared policy methods. All baselines (except for BC and ACER) are from the ConvLab2 framework. Behaviour cloning (**BC**) is a supervised learning method that imitates the oracle actions. Deep Q-network (**DQN**) and policy gradient (**PG**) are standard value based and policy based methods. Actor-critic with experience replay (**ACER**) is a a more sophisticated actor-critic based method (Ziyu Wang et al., 2017). Proximal policy optimization (**PPO**) is an actor-critic method that utilises a surrogate function based on advantage function to update the policy network (Schulman et al., 2017b). Guided dialogue policy learning (**GDPL**) is a joint policy optimization and reward estimation method using adversarial inverse reinforcement learning (Takanobu et al., 2019b).

Method name	Abbrev.
Handcrafted Policy	HDC
Deep Q-Network	DQN
Policy Gradient	PG
Proximal Policy Optimization	PPO
Guided Dialogue Policy Learning	GDPL
Behaviour Cloning	BC
Actor-Critic with Experience Replay	ACER
ACER with Imitation Learning from Oracle Demonstrations	ACER-ILfOD
ACER with Imitation Learning from Oracle Supervision	ACER-ILfOS

Table 3.4: Overview of proposed methods for multi-domain oriented dialogues.

Concerning our propositions, we tested the Imitation Learning from Oracle Demonstrations (ILfOD) method and the Imitation Learning from Oracle Supervision (ILfOS) method based on ACER to study the impact of demonstration and supervision in a multi-domain dialogue context.

Hyper-parameters and Training: All baselines use the default settings proposed by ConvLab2. Concerning our hybrid imitation and reinforcement learning methods, the used oracle is the handcrafted agent proposed by the framework. When we use ILfOD or ILfOS methods, the oracle-agent ratio is preserved $\beta = 0.5$. However, we precise that when we use ILfOS, we call all the time the oracle which gives us the best expert action as supervision.

Our policy algorithm is an off-policy learning that uses experience replay (all data are stored in buffers) without priority, i.e without importance sampling. The

exploitation-exploration procedure is achieved by Boltzmann sampling with a fixed temperature $\tau = 1$ choosing arbitrary. All our proposed networks have the same architecture composed by two hidden layers, both with 128 neurons.

For learning stage, we use a learning rate $\alpha = 10^{-3}$, a dropout rate dr $= 0.1$ and a batch size bs $= 64$. Each loss function has a weight of $\lambda_Q = 0.5$, $\lambda_\pi = 1$, $\lambda_{IL} = 1.$ and $\lambda_{ent} = 0.01$ respectively. We fix a margin penalty $\mu = \log(2)$ for the margin loss. The learning frequency is one iteration after each episode (finished dialogue) with only one gradient iteration.

The system is guided by the rewards as shown in Equation 2.7. At any time, if all domains are solved (a domain is solved if all related tasks are solved), it gains 40 points. On the contrary, if the current active domain is solved, it gains 5 points. Otherwise, it is penalized by 1 point.

Experiments: All the experiments were launched 10 times with random initialisations and the results were averaged. Each learning trajectory was kept up to 10 000 dialogues with a step of 1 000 dialogues in order to analyze the variability and stability of the methods.

We evaluate the performance of the policies for all tasks. For the find task, we use the recall metric called **inform recall rate**. For the book task, we use the accuracy metric called **book rate**. As a reminder (see Chapter 1 for more information), **inform recall rate** evaluates whether all the requested information has been informed. **Book rate** assesses whether the offered entity meets all the constraints specified in the user goal.

The dialogue is marked as successful if and only if both inform recall and book rates are 1. An active domain in dialogue is marked as successful if and only if both inform recall and book rate are 1 in the context of this domain (independently of other domains).

Pre-analysis: It is interesting to note that the distribution of rewards does not reward the system for successfully solving one of two tasks (when two tasks are required). So the system can be encouraged to solve only one of the two possible tasks if they appear alone and being victim of over-learning or partial-learning by ignoring the other task.

In the same way, it is important to notice that success rate (or any other metric) in multi-domain context is not the mean of all success rates in each single-domain context (it is lower)[1]. In fact, if two domains are required, success rate in multi-domain context is the mean of the product of all pairs of success rates in each single-domain context.[2]

[1]The multi-domain-task oriented dialogue problem is more difficult than the single-equivalent because it is harder to success two domains and/or tasks in succession in one dialogue than to success them in two independent dialogues.

[2]Example: Let's flip two coins. On average, we can expect to get 50% heads for the first coin and the same with the second one. But we can only expect to get 25% double heads.

(a) Success rate for reference methods.

(b) Success rate for hybrid IL/RL algorithms.

(c) Recall rate for reference methods.

(d) Recall rate for hybrid IL/RL algorithms.

(e) Book rate for reference methods.

(f) Book rate for hybrid IL/RL algorithms.

Figure 3.7: Distribution of the performance of the proposed approaches with different architectures on multi-domain dialogues in ConvLab, with 10 different initializations. The colored area represents the distribution and the middle line represents its median. *BC* stands for behaviour cloning. *ILfOD* and *ILfOS* stand for imitation learning from oracle demonstrations and from oracle supervision respectively.

Results: Concerning the re-evaluation of the proposed policies in ConvLab, the results are presented in Figures 3.7a, 3.7c and 3.7e. All baselines have difficulties to learn on a horizon of 10 000 dialogues with a clear stagnation of the success rate between 0% and 20%. One of the reasons for this is the difficulty of solving the booking task, exploring efficiently and finding winning trajectories. This is evidence of the difficulties of learning with sparse rewards on multi-domain dialogues.

Then we perform an ablation study based on ACER (in the code, we call this algorithm A2C since it learns the advantage function). The results of this study are presented in Figures 3.7b, 3.7d and 3.7f. The supervised loss-based methods (BC and ILfOS) make more progress in achieving successful dialogues by making a trade-off between finding and booking tasks. We can see an evolution in performance up to 50% for BC and 40% for ILfOS. It can be noted that the use of demonstrations in the RL process alone does not significantly improve performance over the time horizon of the experiment. This clearly shows that demonstrations alone (ILfOD) are not sufficient to achieve good performance in a multi-domain dialogue context, but that

the combined use of supervision and reinforcement is relevant. We can at least notice that ILfOS performs better than BC at the very beginning of the experiment (with 1000 dialogues). In any case, having expert demonstrations is essential to achieve good performance quickly. This observation leads us to wonder whether results are preserved if a small amount of data is available instead of systematically accessing the oracle demonstrations.

3.5 Conclusion

We have presented hybridisation strategies in pure reinforcement learning, in which we learn a stochastic dialogue policy with demonstrations given by a sub-optimal expert designed by humans directly in the learning process. The results of our experiments suggest that this hybridisation approach outperforms classical approaches in noisy environments for task-oriented dialogue systems. Moreover, our results suggest that the demonstrations given by the hand-crafted expert improve performance at the beginning of the learning process without any supervision. Our approach can compete with the hand-crafted expert when classical approaches can not in single-domain dialogue context.

However, this analysis no longer applies when the system learns in a multi-domain dialogue context. All commonly used reinforcement learning methods fail to solve the different tasks simultaneously. Methods based solely on oracle supervision (BC) have been shown to better identify and exploit long-term winning trajectories. On the contrary, the methods based on the hybridisation between oracle demonstration and reinforcement (ILfOS) were more effective in the short term. This finding leads us to wonder whether this conclusion will persist when a small amount of data is available, as opposed to systematic access to oracle demonstrations during the learning process.

Limitations

It would have been interesting to to explore strategies which can help the agent to consult efficiently the demonstrations or the feed-backs of the expert by adding constraints in a similar approach of (L. Chen et al., 2017). We would like to experiment different strategies of how define correctly temperature in Boltzmann sampling when learning with Q-learning in order to adapt these methods to take account of the action space size. Finally, it would have been interesting to train the model with different beta parameters or with a mixture of demonstrations and feed-backs because it would be an important factor in practice to estimate the importance of the diluted expert in the derived policy and so the human cost.

4

Structuring Dialogue Policies with Graph Neural Network

Most of the Reinforcement Learning approaches for DM learns a specific policy for each domain or a more complex overall policy for several domains. Nevertheless, real dialogue systems must handle simultaneously several domains, tasks and slots. Ideally, the DM must detect domain changes, plan across different domains in order to deal with multi-domain dialogues and plan across different slots in order to deal with complex single-domain dialogue stage. However, learning can be difficult because the state-action dimension is larger while the reward signal remains scarce. We present in this Chapter several structured policies based on Hierarchical Reinforcement Learning (HRL) and Graph Neural Network (GNN) that we combine with different degrees of Imitation Learning in order to effectively handle multi-domain dialogues. These methods benefit from strategies that take advantage of the similarities between domains and between slots and the structure imposed by the dialogue problem, which theoretically significantly increase learning efficiency. Our study underline the benefit of structured policies against non structured policies that could accelerate the learning when facing real humans.

4.1 Motivation

We focus here on the multi-domain multi-task dialogue problem. Most of the RL approaches learn a specific dialogue policy for each domain and task, like for instance find a hotel or find a restaurant. However, real applications like personal assistants must deal with multiple tasks: the user may first want to find a hotel (first task), then book it (second task). Moreover, the tasks may cover several domains: the user may want to find a hotel (first task, first domain), book it (second task, first domain), and then find a restaurant nearby (third task, second domain). In Peng et al. (2017) these types of multi-domain dialogues are called *composite*.

A first step to handle this complexity is to rely on a *hierarchy* on domains and tasks: this is called Hierarchical Reinforcement Learning (HRL) (Dayan et al., 1993;

Parr et al., 1998; R. S. Sutton et al., 1999; Dietterich, 2000). Another requirement is *domain scalability* which is the ability of a model to switch from one domain to another by scaling up the policy (Z. Zhang et al., 2020).

In recent works, L. Chen et al. (2018) and Z. Chen et al. (2020a) proposed to improve domain scalability by learning a generic policy (called a *structured policy*), that is based on Graph Neural Network (GNN) (Rahimi et al., 2018; Zhou et al., 2020; Wu et al., 2020). Indeed, the dialogue state spaces and action sets are usually different from a domain to another. However, some generic dialogue actions (*e.g.*, greetings) or some slots (*e.g.*, the hotel/restaurant reservation date) share a similar semantic structure. A key property of GNNs as policy networks is their ability to explicitly wire this knowledge through a graph of communicating sub-networks. This weight-sharing also improves the sparsity of the model with additional benefits on data efficiency, stability, and adaptability.

Although structured dialogue policies as proposed in Z. Chen et al. (2020b) and Z. Chen et al. (2020a) can adapt quickly from a domain to another, covering multiple domains is still harder because the system must detect domain changes and plan across different domains. Therefore, the dimensions of the state and action spaces are larger while the reward signal remains sparse. The combination of HRL and of structured policies is helpful as in Z. Chen et al. (2020b) but is insufficient for truly multi-domain dialogues.

A common technique to circumvent this reward scarcity problem in RL is to guide the learning by injecting some knowledge though a teacher policy. This hybrid imitation and reinforcement learning can be declined at various levels from pure *behaviour cloning* where the agent only learns to mimic its teacher to pure *reinforcement* where no hint is provided (see our previous Chapter 2 and Chapter 3).

Our main contribution in this Chapter is to study how GNN structured policies combined with hybrid IL and RL methods can be effective to handle multi-domain dialogues. We study on the adaptability of GNN policies during the learning and testing stages in ConvLab2 (Zhu et al., 2020). We provide large scale experiments, in which we analyse the performance of different types of policies, from single-domain policy to generic policy, with different RL algorithms and different levels of IL. These experiments underline the clear benefit of structured policies against non-structured policies. Furthermore, we evaluate the best approach in a full dialogue pipeline with simulated and real users to validate structured policies in practice.

For the seek of simplicity and scalability, we propose a hierarchical decomposition of dialogues at multi-domain-level. The objective is to propose a hierarchical structured policy that: 1) adapts to the domain-level, *i.e.* it is scalable to the change in the number of slots and 2) adapts to the multi-domain-level, *i.e.* it is scalable to the change in the number of domains. To the best of our knowledge, an investigation of these methods on true multi-domain dialogues environment has never been proposed.

4.2 Introduction on structured dialogue policies

Fundamental HRL frameworks (Dayan et al., 1993; Parr et al., 1998; R. S. Sutton et al., 1999; Dietterich, 2000) have inspired a wide variety of work on DM. They propose both temporal abstraction (the problem is broken down in time into successive sub-problems) and space abstraction (the problem is partitioned in the state/action space into complementary sub-problems) in decision-making process.

In our work, we used the concept of *hierarchical policy* at the multi-domain level, by decomposing the multi-domain problem into independent single-domain problems (see Subsection 4.2.1). We also used the concept of *structured policy* at the single-domain level, by decomposing the multi-slot problem into interdependent single-slot problems (see Subsection 4.2.2). By breaking down as follows, the initial agent can be considered as a composition of multiple agents. Then, we propose an overview of hierarchical and structured policies applied to DM in Subsection 4.2.3.

4.2.1 Hierarchical policy via Sub-Markov Decision Process

For a mathematical definition of **hierarchical policy**, we refer to the work of R. Sutton et al., 1999 that introduces the concept of Semi-Markov Decision Process (SMDP) using temporal abstraction and to the work of Z. Wen et al., 2020 that introduces the concept of Sub-Markov Decision Process (SubMDP) using state partition. In the scope of our work, we propose to consider the SubMDP framework.

Typically, a **meta-controller** (also called *master policy*) manages a set of controllers which operate at a lower level (we refer them as **sub-policies**). The high-level decision can be taken before the low-level decisions (*i.e.* the action is a choice from a set of sub-policies) or after (*i.e.* the action is a choice from a set of sub-policy actions) as illustrated in Figure 4.1. Therefore, we propose the Definition 4.1:

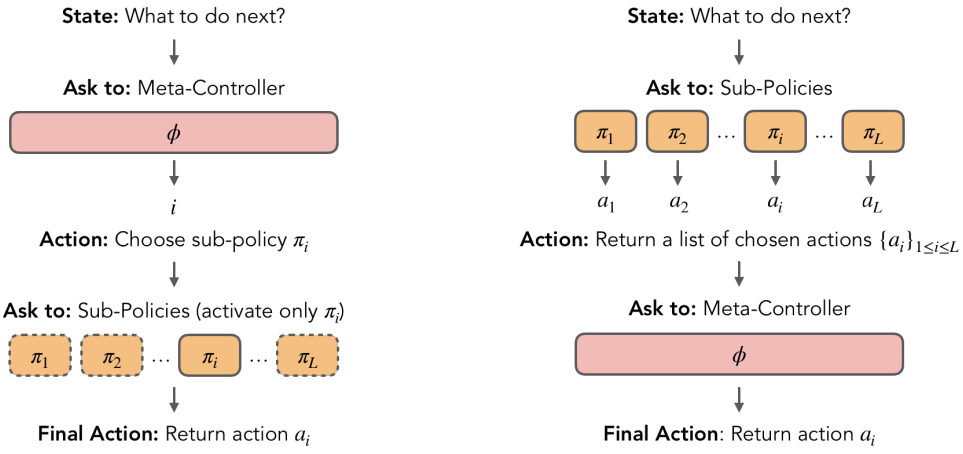


Figure 4.1: Two possible decision processes for a hierarchical policy. On the left, the meta-controller chooses the sub-policy to execute, which will return the final action. On the right, the sub-policies are all executed, returning each one a action, then the meta-controller chooses one of these actions to finally return.

Definition 4.1: Hierarchical policy

A **hierarchical policy** π associated with a MDP $\mathcal{M}$ is defined by a meta-controller ϕ and a set of independent sub-policies π_i, each associated with a sub-problem class of $\mathcal{M}$ (which we will refer to as SubMDP $\mathcal{M}_i$ in the next Definition 4.2).

The meta-controller ϕ identifies in which subMDP the agent is and delegates to the sub-policy π_i the responsibility to solve the corresponding $\mathcal{M}_i$ sub-problem. More formally:

$$\forall s \in \mathcal{S}, \quad \begin{cases} \exists! i \in [1,L] \quad \text{s.t.} \quad s \in \mathcal{S}_i, \\ \pi_i(s) = a_i, \quad \phi(s) = i \quad \text{and} \\ \pi(s) = \pi_{\phi(s)}(s) = \pi_i(s) \end{cases} \tag{4.1}$$

In HRL, a state space partition is assumed to exist if a large problem can be broken down into K sub-problem classes that can be tackled and solved independently (in our example, we can assume that single-domain problems are independent with each other). We formalise the notion of MDP decomposition into sub-problems as following:

Definition 4.2: Sub-Markov Decision Process

Consider a partition of the non-terminal states $\mathcal{S} \setminus \{s_e\}$ into L disjoint subsets $\mathcal{H} = \{\mathcal{S}_i\}_{i=1}^{L}$. An induced **Sub-Markov Decision Process (SubMDP)** $\mathcal{M}_i$ is a sub-process of a MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ based on a partition of the state space $\mathcal{S}$ and formally described by a 4-tuple $\mathcal{M}_i = \langle \mathcal{S}_i \cup \mathcal{E}_i, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i \rangle$ where:

- $\mathcal{M}_i$ is the induced SubMDP of $\mathcal{M}$ restricted to $\mathcal{S}_i$,
- $\mathcal{S}_i$ is the internal state space, $\mathcal{S}_i \cup \mathcal{E}_i$ the state space, $\mathcal{A}$ the action space and $\mathcal{E}_i$ the exit state set such as:

$$\mathcal{E}_i = \{e \in \mathcal{S} \setminus \mathcal{S}_i : \exists (s,a) \in \mathcal{S}_i \times A \quad \text{s.t.} \quad \mathcal{P}(e|s,a) > 0\} \tag{4.2}$$

- $\mathcal{P}_i \in \Delta \mathcal{S}^{\mathcal{S}_i \times \mathcal{A}}$ is the restriction of the transition probability $\mathcal{P}$ to $\mathcal{S}_i \times \mathcal{A}$,
- $\mathcal{R}_i \in \Delta \mathbb{R}^{\mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i}$ is the restriction of the reward function $\mathcal{R}$ to $\mathcal{S}_i \times \mathcal{A}$,
- the SubMDP $\mathcal{M}_i$ terminates once it reaches a state in $\mathcal{E}_i$.

Two SubMDPs $\mathcal{M}_i$ and $\mathcal{M}_j$ are **equivalent** if there exist a bijection such that the SubMDPs have the same internal states, exit states, transition probabilities and rewards at internal states.
Then let $K \leq L$ be the number of equivalence classes of SubMDPs induced by a particular partition $\mathcal{H}$ of $\mathcal{M}$. When there is no repeatable structure, $K = L$. When the partition produces repeatable structure, $K < L$.

In particular, an algorithm based on this kind of structure has theoretical guarantees. As Z. Wen et al. (2020) proves under reasonable assumptions (by extending the work of Osband et al. (2013)), a model-based Thompson sampling-style HRL algorithm that exploits this structure is statistically efficient, as established through a finite-time regret bound analysis in Table 4.1. Posterior sampling for reinforcement learning (PSRL) offers an often effective approach to episodic reinforcement learning (Strens, 2000). PSHRL stands for its hierarchical version. VI stands for Value Iteration planning and PEP for Planning with Exit Profiles planning.

learning alg.	planner	regret bound	computation per episode
PSRL	VI	$\tilde{O}(H^{3/2}\|\mathcal{S}\|\sqrt{\|\mathcal{A}\|T})$	$O(\|\mathcal{S}\|^2\|\mathcal{A}\|M)$
PSHRL	VI	$\tilde{O}(H^{3/2}M\sqrt{K}\|\sqrt{\mathcal{A}\|T})$	$O(\|\mathcal{S}\|^2\|\mathcal{A}\|M)$
PSHRL	PEP	$\Delta\|\mathcal{E}\|T + \tilde{O}(H^{3/2}M\sqrt{K}\sqrt{\|\mathcal{A}\|T})$	$O(X(M^2K\|\mathcal{A}\| + \|\mathcal{E}\|^2)M)$

Table 4.1: Differences in regret bounds and computational complexities for (non)-hierarchical algorithms taken from Z. Wen et al. (2020). H is a bound on the expected time horizon of $\mathcal{M}$; T is the number of interaction episodes; M is the maximum SubMDP size; K is the number of SubMDP equivalence classes; $\mathcal{E}$ is the set of all exit states. ; Δ and X respectively measure the quality and the number of exit profiles used in PEP.

More specifically, HRL will offer dramatic improvements over the standard algorithms only if $\mathcal{M}$ exhibits hierarchical structure, in particular if $MK << |\mathcal{S}|$ and $|\mathcal{E}| << |\mathcal{S}|$ where M is the maximum size of an induced SubMDP, K the number of SubMDP equivalence classes and $|\mathcal{E}|$ the total number of exit states.

With regard to our work, we address the problem of multi-domain task-oriented dialogue by proposing a hierarchical policy: the state space is partitioned according to the domains. The meta-controller identifies the currently activated domain. The sub-policies are responsible for solving the task-oriented dialogue problem specific to their respective domain. When there are L balance domains with equivalent tasks, $M = |\mathcal{S}|/L$, $K = 1$ and thus $MK = |\mathcal{S}|/L$.

4.2.2 Structuring policies with interdependence graph

A state space partition is assumed to exist if a large problem can be broken down into sub-problems. But sometimes, they can not be tackled and solved independently. In our example, in a single-domain problem with multiple slots, we assume that we can not manage the slots independently. Indeed, a sub-policy with a specific view on a slot may need information on other slots (*e.g.* it is not relevant to ask the user to book a hotel room until the date is known). The problem of single-domain task-oriented dialogue can not be handled sufficiently well with hierarchical policy. However, it is still possible to build a hierarchical policy with non-independent sub-policies, namely **structured policy** as a generalisation of hierarchical policy.

Definition 4.3: Structured policy

A **structured policy** π associated with a MDP $\mathcal{M}$ is defined by a meta-controller ϕ and a set of interdependent sub-policies π_i, each associated with a SubMDP $\mathcal{M}_i$.

The sub-policies π_i are allowed to communicate with each other before planning a strategy for their respective $\mathcal{M}_i$ sub-problem. The meta-controller ϕ is responsible for consulting all sub-policies to solve the main problem $\mathcal{M}$. More formally:

$$\forall s \in \mathcal{S}, \quad \begin{cases} \exists! i \in [1,L] \quad \text{s.t.} \quad s \in \mathcal{S}_i, \\ \pi_i(s|\{\pi_j, \forall j \neq i\}) = a_i, \quad \phi(s|\{\pi_j, \forall j \in [1,L]\}) = i \quad \text{and} \\ \pi(s) = \pi_{\phi(s)}(s) = \pi_i(s) \end{cases} \tag{4.3}$$

With regard to our work, we address the problem of single-domain task-oriented dialogue by proposing a structured policy: the **state features space** is partitioned according to the slots such that the sub-agents communicate with each other the information they have on their respective slots. The meta-controller consults the sub-agents and chooses the best one to play the next action.

While a hierarchical policy (with independent sub-policies) can be seen as a multi-agent policy with one active sub-policy per turn, a structured policy (with interdependent sub-policies) can be seen as a multi-agent policy in which these sub-agents cooperate with each other to find out which one is best to take the next action. Figure 4.2 highlights the structural differences between hierarchical and structured policies.

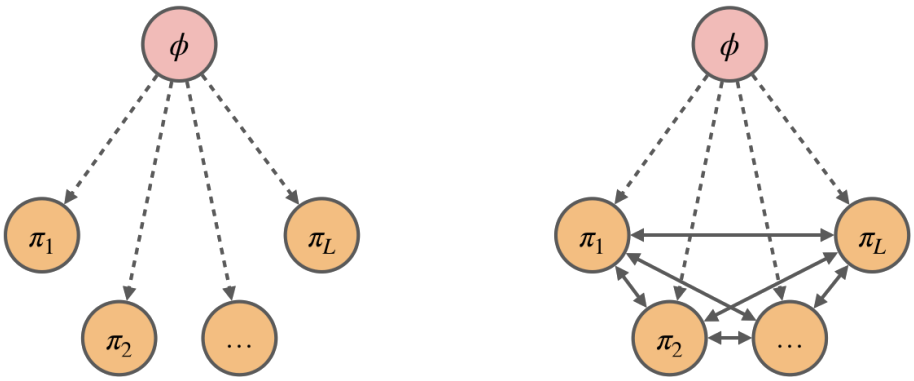


Figure 4.2: Structural comparison between hierarchical policy (on the left) and structured policy (on the right) according to Definitions 4.1 and 4.3. The dotted lines represent the subordination of the sub-policies to the meta-controller. The solid lines represent the interdependence or communication between sub-policies. In hierarchical policy, sub-policies are not influenced by others in their decision-making process. In structured policy, the sub-policies influence each other.

Where graphs are all around us

This structured organisation can be naturally represented quite naturally as graphs. We can find them all around us as graphs are an extremely powerful and general representation of data: images, texts, molecules, social or citation networks (Sanchez-Lengeling et al., 2021). More formally in the Definition 4.4:

Definition 4.4: Graph structure

A **graph** $\mathcal{G} = (V, E, U)$ is a data structure representing relations E (edges) between a collection of entities V (nodes). To further describe a graph, information can be stored in:

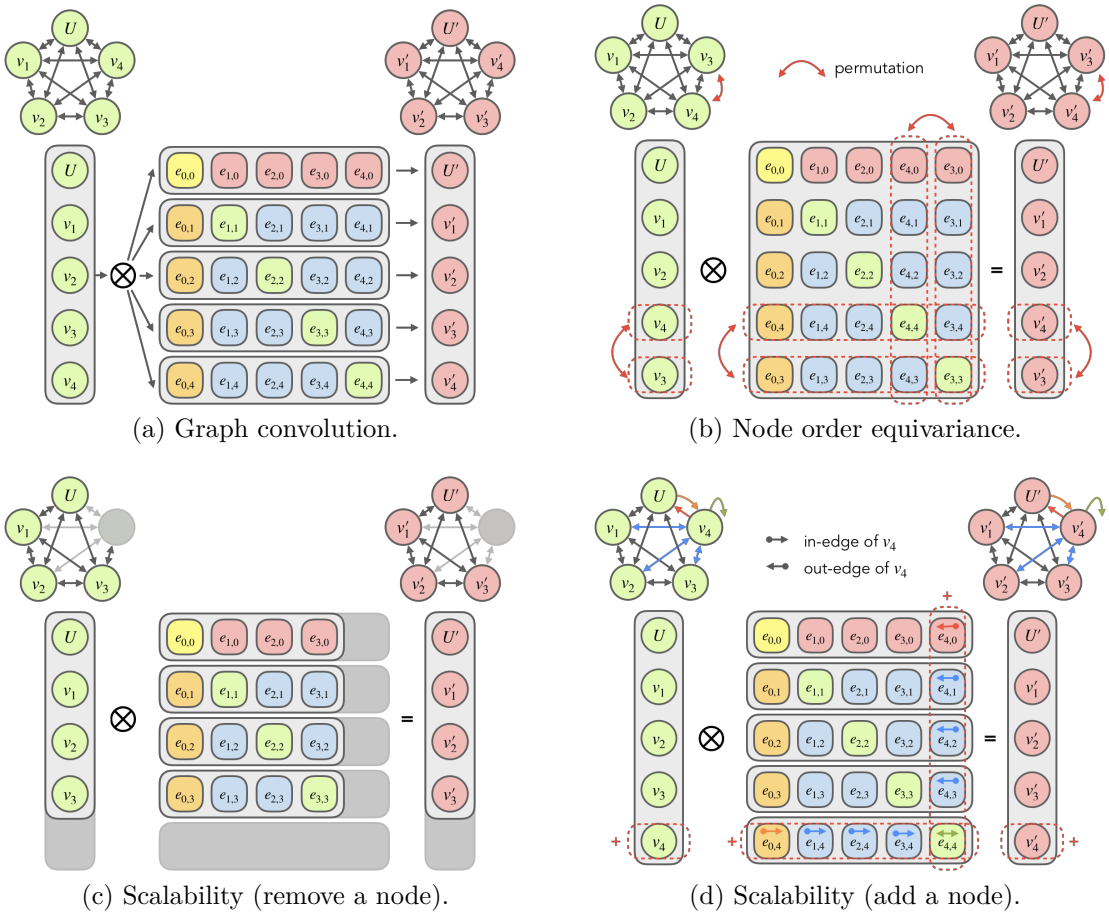
- the vertex (nodes) attributes in a set of vectors $V = \{v_i\}_{i=1}^{L}$,
- the edge (link) attributes and directions in a set of vectors or an adjacency matrix $E = \{e_{i,j}\}_{i,j=1}^{L}$,
- the global (or master node) attributes in one vector U.

There are many useful problems that can be formulated over graphs: *node, edge and graph classification* where nodes, edges or graph are individually classified, *node clustering* where similar nodes are grouped together based on connectivity, *link prediction* where missing links are predicted and *influence maximisation* where influential nodes are identified (Daigavane et al., 2021).

For over a decade, researchers have developed neural networks that operate on this kind of graph data called Graph Neural Network (GNN) (Scarselli et al., 2008). These methods are increasingly used in areas such as antibacterial discovery (Stokes et al., 2020), physics simulations (Sanchez-Gonzalez et al., 2020), fake news detection (Monti et al., 2019), traffic prediction (Jiang et al., 2022) and recommendation systems (Eksombatchai et al., 2018).

A GNN is an optimisable transformation on all attributes of the graph consisting of a cascade of layers, each of which applies a graph convolution, followed by a point-wise non-linearity (Daigavane et al., 2021). In particular, it preserves graph symmetries also called **node-order equivariance** or **permutation invariance** property (Gama et al., 2020): as illustrated in Figure 4.3b, if we permute the nodes in a certain way, the resulting representations of the nodes should also be permuted in the same way. It also benefits from the ability to adapt to the structure of the data also called the **scalability** property: as in Figures 4.3c and 4.3d, if we add or remove nodes in a certain way, the network architecture is scaled in the same way.

A growing number of GNN architectures are constantly being proposed. Graph Convolutional Networks (GCN) applies a normalised graph convolution (Kipf et al., 2016), Graph Attention Networks (GAT) uses a multi-headed attention mechanism (Veličković et al., 2017), Graph Sample and Aggregate (GraphSAGE) is based on neighbourhood sampling and aggregation (Hamilton et al., 2017) and Graph Isomorphism Network (GIN) has more discriminative power (Xu et al., 2018).



(a) Graph convolution.

(b) Node order equivariance.

(c) Scalability (remove a node).

(d) Scalability (add a node).

Figure 4.3: Permutation invariance and scalability properties in GNN. The green nodes are the input of the GNN and the red ones the output. The $\otimes$ symbol represents the graph convolution. The colored squared represent the weights of the GNN with respect to the adjacency matrix (same color, same weight). A row in the adjacency matrix represents the "all nodes to one node" relationship. A permutation of the input corresponds to a permutation of the rows and columns in the matrix and therefore resulting in a permutation of the output. The addition or deletion of a node corresponds to the addition or deletion of the weights of GNN according to the adjacency matrix. We assume here that all nodes are of the same type and that their link is of the same type.

4.2.3 Related Work

Hierarchical Reinforcement Learning for Dialogue Manager

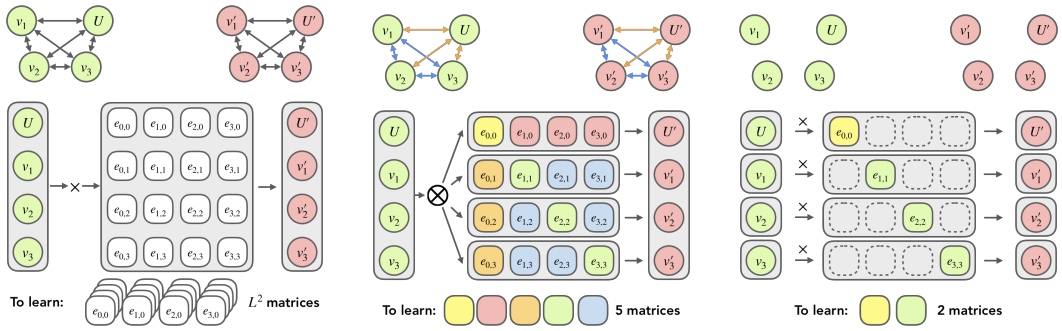
In single-domain task-oriented dialogue systems, HRL is used by DM to delegates the low-level dialogue actions to sub-policies, to share slot management between them. For instance, the *feudal hierarchy* has been combined with Actor Critic with Experience Replay (ACER) and with Deep Q-Network (DQN) by dividing the main task into sub-tasks and the master policy's action is to choose a slot-dedicated sub-policy (Casanueva et al., 2018a; Casanueva et al., 2018b). These sub-policies are based on the Domain Independent Parametrisation (DIP) that allows for weight

sharing between sub-policies (Zhuoran Wang et al., 2015; Papangelis et al., 2019).

DIP (Zhuoran Wang et al., 2015) standardises the representations of states and actions into a common feature space to build policies that can be transferable between domains. It has been used to compare GP-SARSA and GP-SARSA-DIP (Zhuoran Wang et al., 2015), then DQN and DQN-DIP (Papangelis et al., 2019), It has also been used to compare DQN-DIP and Feudal-DQN-DIP (Casanueva et al., 2018a), then ACER-DIP and Feudal-ACER-DIP (Casanueva et al., 2018b).

Structured policies for Dialogue Manager

A more structured hierarchy with GNN rather than a set of classical Feed-forward Neural Network (FNN) allows one to build non-independent sub-policies resulting in a more expressive global policy. This decomposition of decisions appears to be a generalization of DQN-DIP methods and a restriction of DQN methods (as shown in Figure 4.4). Another major advantage of structured policies is their scalability and thus their ability to generalize between similar dialogue tasks. GNN has been used in various works to compare DQN-DIP and DQN-GNN-DIP (L. Chen et al., 2018) and its ACER variant (Z. Chen et al., 2020a).



(a) DQN-FNN. (b) DQN-DIP-GNN. (c) DQN-DIP.

Figure 4.4: Comparison of the internal structure of neural networks. (a) is a FNN (as used in DQN). Its functional space is very complex because all sub-matrices (in white) need to be learned. The $\times$ symbol represents the matrix multiplication. (b) is a GNN (as used in DQN-DIP-GNN). Its functional space is less complex because it relies on symmetries (the color sub-matrices share the same weights) and only 5 matrices need to be learned in this example (the total number of learned matrices depends on the number of different types of links that exist in the graph). The $\bigotimes$ symbol represents the graph convolution. We assume here that all nodes are of the same type and that their link is of the same type. (c) is a hierarchical FNN (as used in DQN-DIP). Its functional space is the least complex because only 2 matrices need to be learned (the total number of learned matrices depends on the number of different types of nodes that exist in the graph). As a counterpart, the sub-policies are independent. The dotted squares mean that there is no matrix. The symbol $\times$ therefore represents the matrix multiplication only between a node and a sub-matrix. We assume here that all nodes are of the same type.

State of the art

These hierarchical and structured mechanisms have been used in recent methods: FeudalGain (Geishauser et al., 2021) is based on Feudal-ACER with an intrinsic reward based on information gain and STRAC (Z. Chen et al., 2020a) is an ACER variant of DQN-GNN that proposes an implicit policy decomposition and uses noisy networks for exploration. The purpose of these two methods was to improve domain scalability and learning efficiency. With ComNet, Z. Chen et al. (2020b) were the first to propose a multi-domain dialogue variant of DQN-GNN by adding a domain-selection hierarchy level. For experimental purposes, they discarded the domain provided by the Dialogue State Tracking (DST) module. They relied on the DIP to share the same GNN structure for all domain-specific sub-policies and they evaluated their policies in PyDial (Ultes et al., 2017b).

4.3 Hierarchical and structured policies for multi-domain dialogues

In this work, as in L. Chen et al. (2018) and Z. Chen et al. (2020a), we propose to improve multi-domain covering by learning a generic policy based on GNN. But unlike them, (i) we use a multi-domain multi-task setting, in which several domains and tasks can be evoked in a dialogue; (ii) the DST output is not discarded when activating the domain; and (iii) we adapt the GNN structure to each domain by keeping the relevant nodes while sharing the edge's weights.

First of all, we need to talk about the DIP representation which is the basis of structured dialogue policies in Subsection 4.3.1. Next, we go into the mechanism of GNN and their properties in Subsection 4.3.2. Finally, in Subsection 4.3.3 we propose our hierarchical decomposition of dialogues at multi-domain-level and structured and scalable dialogue policy at single-domain-level.

4.3.1 Data structure: Domain Independent Parametrisation

We adopt the Domain Independent Parametrisation (DIP) state and action representations which standardises the slots representation into a common feature space (Zhuoran Wang et al., 2015). More specifically, these representations are not reduced to a flat vector but to a set of sub-vectors: one corresponding to the domain parametrisation (or *slot-independent representation*), the others to the slots parametrisation (or *slot-dependent representations*) as shown in Figure 4.5.

For any active domain, the input to the *slot-independent representation* is the concatenation of the previous *slot-independent* user and system actions (see examples of the output below), the number of entities fulfilling the user's constraints in the database, the booleans indicating if the dialogue is terminated and whether an offer has been found / booked. The output corresponds to action scores such as

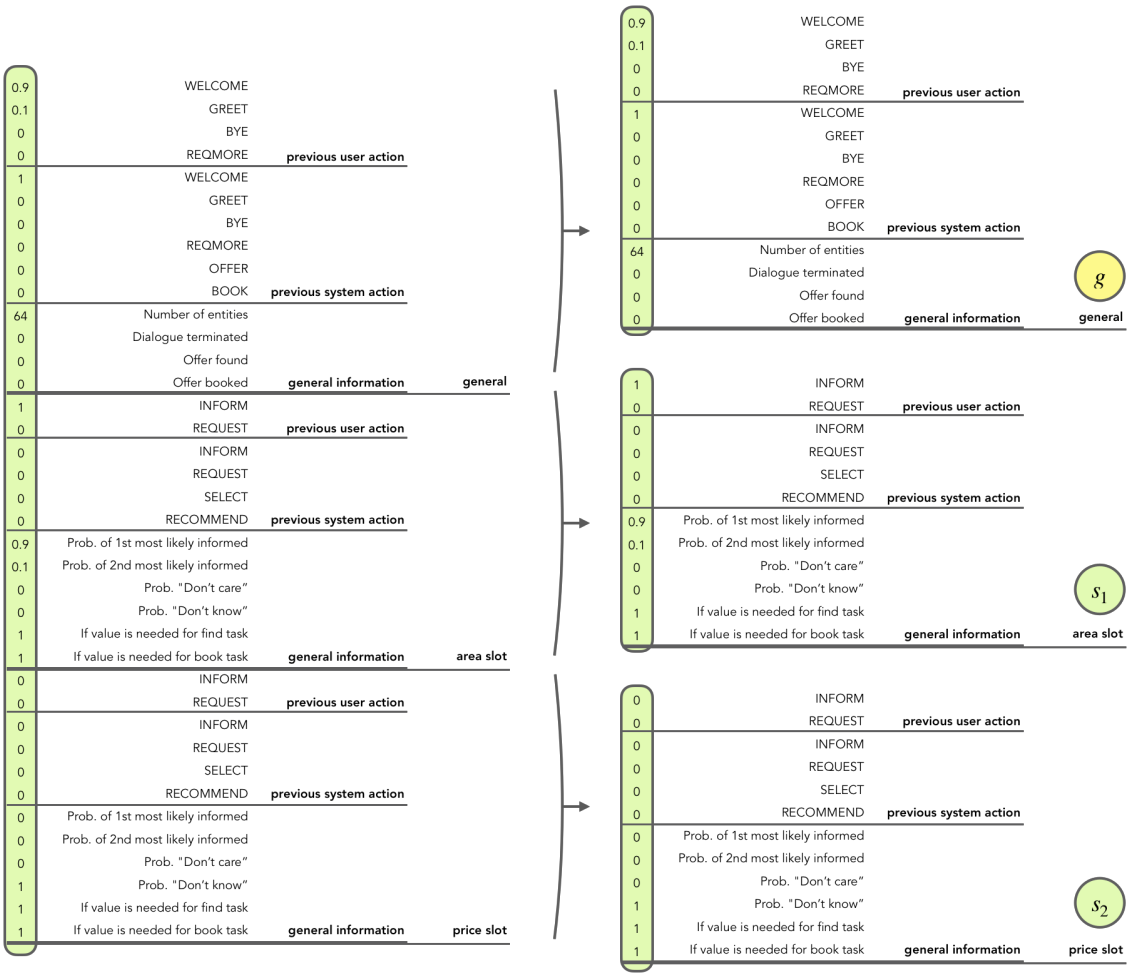


Figure 4.5: Domain Independent Parameterisation (DIP). For any domain, DIP decomposes the summary representations of state and action (the state parameterisation is illustrated here) into a set of sub-vectors. The *slot-independent vector* groups together all the information that does not depend on any slot. It is represented here by a yellow node. For any slot, the called *slot-dependent vector* groups all the information which depends on this slot. It is represented by a green node. All the *slot-dependent representations* have the same parameterisation which allows a common parameterisation between all the slots of all the domains. The example shows the parameterisation of a domain composed of two slots, "price" and "area".

REQMORE, OFFER, BOOK, GREAT, etc.

Regarding the *slot-dependent representation*, its input is composed of the previous *slot-dependent* user and system actions (see output below), the booleans indicating if a value is known and whether the slot is needed for the *find* / *book* tasks. Its output are actions scores such as INFORM, REQUEST and SELECT.

As a restriction imposed by the framework, the parameterisation used depends on the representation of the deterministic states of ConvLab2 which does not consider the uncertainty in the predictions made by the Natural Language Understanding (NLU) module. Ideally, it should be otherwise and we will therefore propose an analysis of the robustness of the tested methods in a noisy environment.

4.3.2 Architecture: Graph Neural Network

Until now, we have always assumed that structured methods allow to express more expressive ans scalable policies thanks to a communication graph. We propose first to formalise this communication graph, then to give the technical details for their implementation in a GNN-type architecture.

How to implement interdependence between sub-policies?

We have mentioned that it is possible to partition the state feature space. To be precise, this partition induces that the sub-policies share different point of view of the large problem. In the multi-domain level, each sub-policy sees only the state of its domain. In the single-domain level, each sub-policy sees only the state features of its slot. Thus, this state partition is only an abstraction of the decomposition of the initial agent into multiple sub-agents as we wish to illustrate in Figure 4.6.

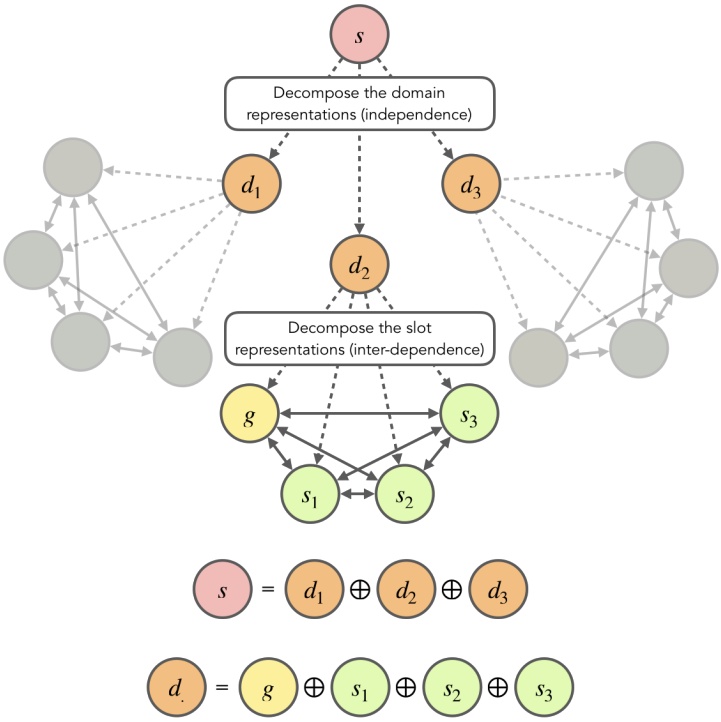


Figure 4.6: Partition of the state feature space at multi-domain and single-domain level (also works for the action feature space). The full state (s) is decomposed according to the domains (from d_1 to d_3). We assume that the domain sub-policies are independent in our work. Then, each domain state is decomposed in several slot states according to DIP (from g to s_3). We assume that the slot sub-policies are inter-dependent in our work. The $\bigoplus$ symbol represents the concatenation.

From this observation, a standard Feed-forward Neural Network (FNN) can be used as model for a structured policy. Indeed, the state space partition induces a weight matrix partition, and each sub-matrix can be considered as a communication module from one sub-agent to another like an adjacency matrix (see Figure 4.6 and the comparison between FNN and GNN in Figure 4.4).

So far, we have never talked about the assumptions about these communication modules. Should the communications be the same between sub-agents? Should they be enabled or disabled? What type of messages should be sent? How should they be aggregated? In fact, the relationships between sub-policies can be formalised by a graph. This justifies the use of Graph Neural Network (GNN) for modelling a structured policy as we will see (as illustrated in Figure 4.6).

What is the formal definition of the communication graph?

We refer to the works of Z. Chen et al. (2020a) to describe the "graph" of communication between slot sub-agents. Here we propose the following definition:

Definition 4.5: Communication graph

A **communication graph** is a fully connected and directed graph $\mathcal{G} = (V, E)$ consisting on a finite set of sub-agents $V = \{v_i\}_{i=0}^{L}$ and relationships between all sub-agents $E = \{e_{i,j}\}_{i,j=0}^{L}$.

- Each node of the graph v_i denotes a sub-agent with sub-policy π_i.
- A directed edge between two sub-agents of the graph $e_{i,j}$ means that the starting sub-agent v_i sends some message to the ending sub-agent v_j.

Related to the Definition 4.4, the master node is treated as a node v_0 called *general node*. The others are called *slot nodes*.

Each sub-agent v_i has a specific view on the current state $\mathcal{S}_i \subset \mathcal{S}$ and can take specific actions $\mathcal{A}_i \subset \mathcal{A}$ depending on the partition of the state/action space .

In particular, for a single-domain problem, we assume that the specific-slot sub-agents share the same policy except for the non-slot sub-agent. This exception is just a convention to handle global information about the conversation that is not related to any slot. We thus identify two roles for sub-agents: the first role called **I-agent** is associated with the non-slot sub-agent v_0 responsible of slot-independent decision making; the second role called **S-agent** is associated with the other sub-agents $\{v_i\}_{i\neq 0}$ responsible of slot-dependent decision making.

Consequently, we identify several types of connection between sub-agents: the first connections called **self-I relation** and **self-S relation** correspond to the connection of an sub-agent with itself ($\{e_{i,i}\}_{i=0}$ is a self-I relation and $\{e_{i,i}\}_{i\neq 0}$ are self-S relations); the other connections called **I2S**, **S2I** and **S2S relations** correspond to the connection between two different sub-agents ($\{e_{i,j}\}_{i=0,j\neq 0}$ are relations from I-agent to S-agent, $\{e_{i,j}\}_{i\neq 0,j=0}$ are relations from S-agent to I-agent and $\{e_{i,j}\}_{i,j\neq 0,i\neq j}$ are relations from S-agent to another S-agent). We omit the I2I relation because only one sub-agent has the I-agent role.

FNN: First step in understanding the GNN structure

The baseline we consider until now is a classical Feed-forward Neural Network (or multi-layer perceptron). The dialogue state is built in such a way that all information is contained in a single input vector. Similarly, the probabilistic distribution over dialogue actions is based on a single output vector. To draw a parallel with DIP, it is as if all the slot-dependent and slot-independent representations had been concatenated into a single vector.

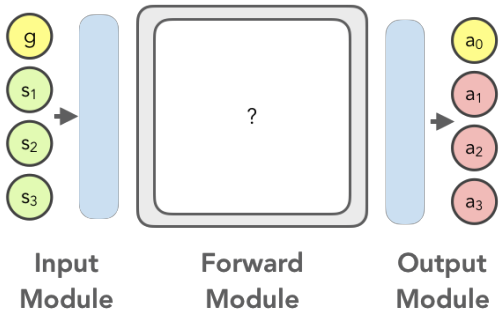


Figure 4.7: FNN module decomposition.

Let s be the dialogue state, $\mathbf{x}$, $\mathbf{h}^l \forall l \in [0, L-1]$ and $\mathbf{y}$ be respectively the input, hidden and output dialogue representations, ϕ be the feature function and $F^l, \forall l \in [0, L-1]$ be the network layers with non-linear functions σ^l parametrised by the weight matrices $\mathbf{W}^l$ and bias vectors $\mathbf{b}^l$.

Input Module

$$\mathbf{h}^0 = \phi(\mathbf{s}) = \mathbf{x} \tag{4.4}$$

Hidden Module

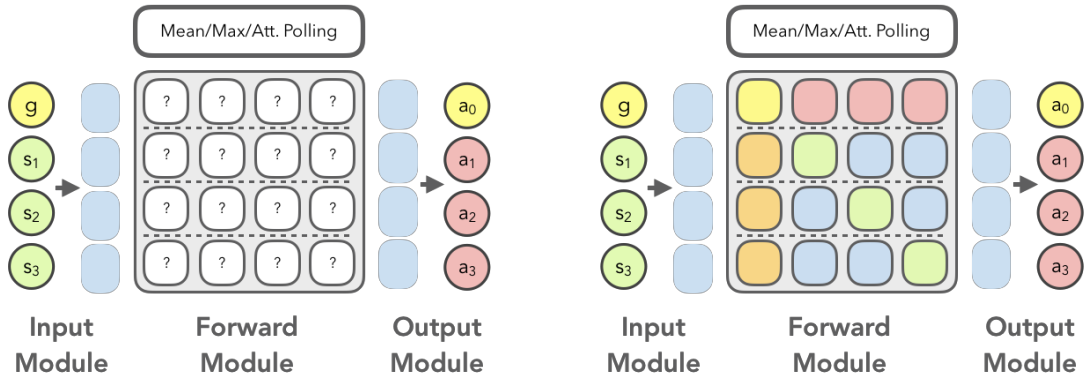
$$\forall l \in [1, L-1], \quad \mathbf{h}^l = F^l(\mathbf{h}^{l-1}) = \sigma^l(\mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l) \tag{4.5}$$

Output Module

$$\mathbf{y} = \sigma^L(\mathbf{W}^L \mathbf{h}^{L-1} + \mathbf{b}^L) \tag{4.6}$$

GNN: The core mechanism of graph convolution

Let us consider a communication graph as presented in Definition 4.5. The dialogue state with DIP is considered as a full-connected graph in which the nodes are the slots according to the decomposition of slot-dependent and independent states. So by construction, the probabilistic distribution over actions is computed by the nodes in a such a way that each slot node is responsible to its slot actions. This abstraction is only a way to model relations between slots and sub-policies and so the use of symmetries in the NN architecture via an adjacency matrix (see Figure 4.8).



(a) Without symmetry assumption (b) With symmetry assumption

Figure 4.8: GNN module decomposition.

We propose here the first steps for modelling GNN architecture. Let s be the dialogue state, $\mathbf{x}_0$, $\mathbf{h}_0^l \, \forall l \in [0,L-1]$ and $\mathbf{y}_0$ be the successive slot-independent representations and $\forall i \in [1,n]$, $\mathbf{x}_i$, $\mathbf{h}_i^l \, \forall l \in [0,L-1]$ and $\mathbf{y}_i$ the successive slot-dependent representations, ϕ_i be the feature function and $\forall i \in [1,n], F_i^l, \forall l \in [0,L-1]$ be the network layers with non-linear functions σ^l parametrised by the weight matrices $\mathbf{W}_i^l$ and bias vectors $\mathbf{b}_i^l$. The proposed GNN is based on a Graph Convolutional Network (GCN) as formalised in Equations 4.7, 4.8 and 4.9 (Kipf et al., 2016):

Input Module

$$\forall i \in [0,n], \quad \mathbf{h}_i^0 = F_i^0(\phi_i(s)) = F_i^0(\mathbf{x}_i) = \sigma^0(\mathbf{W}_i^0 \mathbf{x}_i + \mathbf{b}_i^0) \tag{4.7a}$$

Hidden Module (graph convolution with generic aggregation)

$$\forall l \in [1,L-1], \quad \forall i \in [0,n], \quad \mathbf{h}_i^l = F_i^l(\mathbf{h}^{l-1}) \tag{4.8a}$$

$$\text{Message Sending :} \quad \mathbf{m}_{i \leftarrow j}^l = M_{i \leftarrow j}^l(\mathbf{h}_j^{l-1}) = \mathbf{W}_{i \leftarrow j}^l \mathbf{h}_j^{l-1} \tag{4.8b}$$

$$\text{Message Aggregation :} \quad \mathbf{m}_i^l = A_i^l(\mathbf{m}_{i \leftarrow *}^l) \tag{4.8c}$$

$$\text{Representation Update :} \quad \mathbf{h}_i^l = U^l(\mathbf{m}_i^l) = \sigma^l(\mathbf{m}_i^l) \tag{4.8d}$$

Output Module

$$\mathbf{y} = \sigma^L \left(\bigoplus_{i=1}^{n} \mathbf{W}_i^L \mathbf{h}_i^{L-1} + \mathbf{b}_i^L \right) \tag{4.9}$$

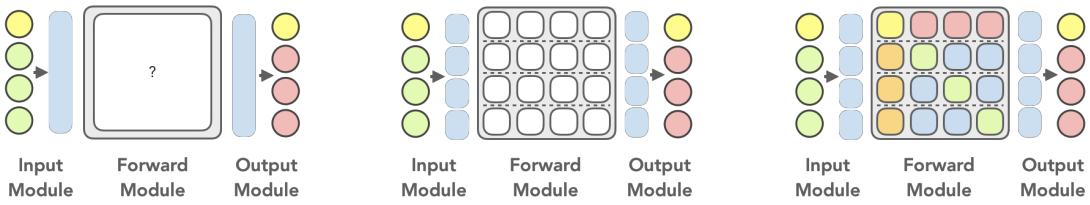
The notation $i \leftarrow j$ is to denote a message sending from sub-policy π_j to sub-policy π_i. It also corresponds to the directed relation between the slots j and i in the adjacency matrix of the communicative graph. By extension, the notation

$\mathbf{m}^l_{i\leftarrow *}$ is to denote the set of all messages sending from sub-policies π_j with $j \neq i$ to sub-policy π_i. The concatenation notation ($\oplus$ symbol) is to denote that all hidden slot representations are regrouped in one single vector.

The graph convolution is detailed in the hidden module (Equations 4.8) and proceeds as follows: the **message sending** functions $M^l_{i\leftarrow j}$ (Equation 4.8b) sends a message from π_j to π_i and could be a linear transformation possibly offset by some bias (we will omit it for easy reading). The **message aggregation** function A^l_i (Equation 4.8c) aggregates all messages received for π_i and could be the average pooling function, the max pooling function or the self-attention mechanism. The **representation update function** U^l (Equation 4.8d) compute the new hidden representation of any sub-policy π_i with non-linear activation function.

GNN vs. FNN: An unified framework to bridge the gap between dense and graph-based architecture

As illustrated in Figure 4.9, the FNN architecture (4.9a) can be seen as a non-flexible and agnostic GNN architecture (4.9c). The power of the GNN architecture is revealed when sub-agents share the same role and therefore share network parameters (as illustrated with the color coding in Figure 4.9c).



(a) Classical FNN repr. (b) FNN repr. with DIP. (c) GNN repr. with DIP.

Figure 4.9: Structures of the neural network layers. The central box represents the weight matrix of a layer. It can be decomposed into sub-matrices. The left circles represent the state vectors of the slots as the input of the network (it can be the concatenation of all the vectors of the slots or a collection of sub-vectors). The right circles represent the action vectors of the slots at the output of the network.

In our single-domain task-oriented dialogue problem, all slot-related sub-agents share the same sub-policy: $\forall i \in [1,n], \pi_S = \pi_i$. In consequence, they share the same message sending and aggregation functions and thus the same weight parameters: $\forall i,j \in [1,n], \mathbf{W}_{i\leftarrow j} = \mathbf{W}_{j\leftarrow i}$ (blue sub-matrices), $\mathbf{W}_{i\leftarrow i} = \mathbf{W}_{j\leftarrow j}$ (green sub-matrices), $\mathbf{W}_{0\leftarrow i} = \mathbf{W}_{0\leftarrow j}$ (red sub-matrices) and $\mathbf{W}_{i\leftarrow 0} = \mathbf{W}_{j\leftarrow 0}$ (orange sub-matrices).

From this point of view, the FNN architecture can therefore be seen mathematically as a GNN architecture where all sub-blocks are not constrained to be identical. We propose a mathematical framework to understand FNN hidden module as a graph convolution in Equations 4.11 and to visualise the decomposition of a weight matrix in a hidden layer of a FNN in Equation 4.10. The element $\mathbf{W}^l_{[i,j]} = \mathbf{W}^l_{i\leftarrow j}$ denotes the sub-block of index i and column j of the matrix $\mathbf{W}^l$.

FNN hidden module seen as a decomposition of the weight matrix

$$\forall l \in [1, L-1], \quad \forall i \in [0, n], \quad \mathbf{h}^l = F^l(\mathbf{h}^{l-1}) = \sigma^l\left(\mathbf{W}^l \mathbf{h}^{l-1}\right) \tag{4.10}$$

$$\text{with} \quad \mathbf{W}^l = \begin{bmatrix} \mathbf{W}^l_{0\leftarrow 0} & \mathbf{W}^l_{0\leftarrow 1} & \cdots & \mathbf{W}^l_{0\leftarrow n} \\ \mathbf{W}^l_{1\leftarrow 0} & \mathbf{W}^l_{1\leftarrow 1} & \cdots & \mathbf{W}^l_{1\leftarrow n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}^l_{n\leftarrow 0} & \mathbf{W}^l_{n\leftarrow 1} & \cdots & \mathbf{W}^l_{n\leftarrow n} \end{bmatrix} \quad \text{and} \quad \mathbf{h}^l = \begin{bmatrix} \mathbf{h}^l_0 \\ \mathbf{h}^l_1 \\ \cdots \\ \mathbf{h}^l_n \end{bmatrix}$$

FNN hidden module seen as a graph convolution

$$\forall l \in [1, L-1], \quad \forall i \in [0, n], \quad \mathbf{h}^l_i = F^l_i(\mathbf{h}^{l-1}) = \sigma^l\left(\sum_{j=0}^{n} \mathbf{W}^l_{i\leftarrow j} \mathbf{h}^{l-1}_j\right) \tag{4.11a}$$

$$\text{Message Sending :} \qquad \mathbf{m}^l_{i\leftarrow j} = M^l_{i\leftarrow j}(\mathbf{h}^{l-1}_j) = \mathbf{W}^l_{i\leftarrow j} \mathbf{h}^{l-1}_j \tag{4.11b}$$

$$\text{Message Aggregation :} \qquad \mathbf{m}^l_i = A^l_i(\mathbf{m}^l_{i\leftarrow *}) = \sum_{j=0}^{n} \mathbf{W}^l_{i\leftarrow j} \mathbf{h}^{l-1}_j \tag{4.11c}$$

$$\text{Representation Update :} \qquad \mathbf{h}^l_i = U^l(\mathbf{m}^l_i) = \sigma^l\left(\sum_{j=0}^{n} \mathbf{W}^l_{i\leftarrow j} \mathbf{h}^{l-1}_j\right) \tag{4.11d}$$

In this case, the use of GNN is only relevant if symmetries exist in the graph structure and allow for weight sharing. The symmetries that are at work on sub-blocks $\mathbf{W}_{i\leftarrow j}$ depend on the assumptions made on the adjacency matrix as illustrated in Figure 4.10. In our case, because all slot-related sub-agents share the same sub-policy, we assume that the graph structure is **slot-node permutation invariant**.

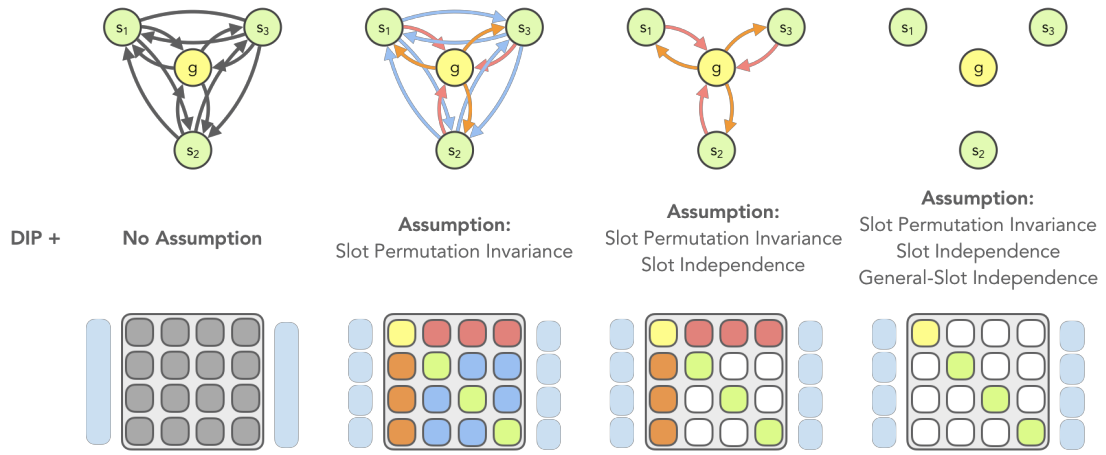


Figure 4.10: GNN and its decomposition variants.

4.3.3 Model: Hierarchical and structured dialogue policy

In our work, we propose a hierarchical decomposition of dialogues at multi-domain-level and a structured decomposition at single domain-level (as illustrated in Figure 4.11). In particular:

- at the multi-domain level, we propose a **domain-selection module** to implement a **hierarchical policy** with one equivalence class for all domains. A meta-controller ϕ chooses the current domain to activate and calls its associated sub-policy. It is trivially aligned with the prediction of the DST module. We assume that all sub-policies are represented by the same sub-policy π for all domains. A direct consequence is that the meta-controller ϕ is scalable to the changes of the number of domains (as far as the DST module is also scalable to theses changes).
- at a single domain level, we propose a **domain-specific decision module** to implement a **structured policy** π with an unique sub-policy, denoted as *I-agent*, and as many sub-policies as there are slots in the activated domain, denoted as *S-agents*. We assume that these interdependent sub-policies π_i communicate with each other via a communication graph that we can implement with a GNN built according to their role, I-agent and S-agent. A direct consequence is that the structured policy π is scalable to the changes of the number of slots (as long as the slots are set by DIP).

By doing that, we propose to improve multi-domain covering by learning a generic policy π for all domains based on GNN. But unlike Z. Chen et al. (2020b), (i) we use a multi-domain multi-task setting, in which several domains and tasks can be evoked in a dialogue; (ii) the DST output is not discarded when activating the domain; and (iii) we adapt the GNN structure to each domain by only keeping the relevant nodes (but, as we will see furthermore, the edge's weights remain shared).

Domain-selection module at multi-domain level

Our method at the multi-domain level works as follow: first, the DST domain-selection module chooses which domain to activate, then, the multi-domain state and action space is projected into the active domain (*i.e.* only the DIP nodes corresponding to the active domain are kept).

Domain-specific decision module at single-domain level

Afterwards at the single-domain level, we apply the GNN message passing but only among the domain specific DIP nodes. Concerning the structured policy at a single-domain-level, we refer to our previous explanations on communicative graph and GNN architecture. We recall that, by construction, the probabilistic distribution over the dialogue actions is computed by the nodes in such a way that each slot node is responsible of its own actions.

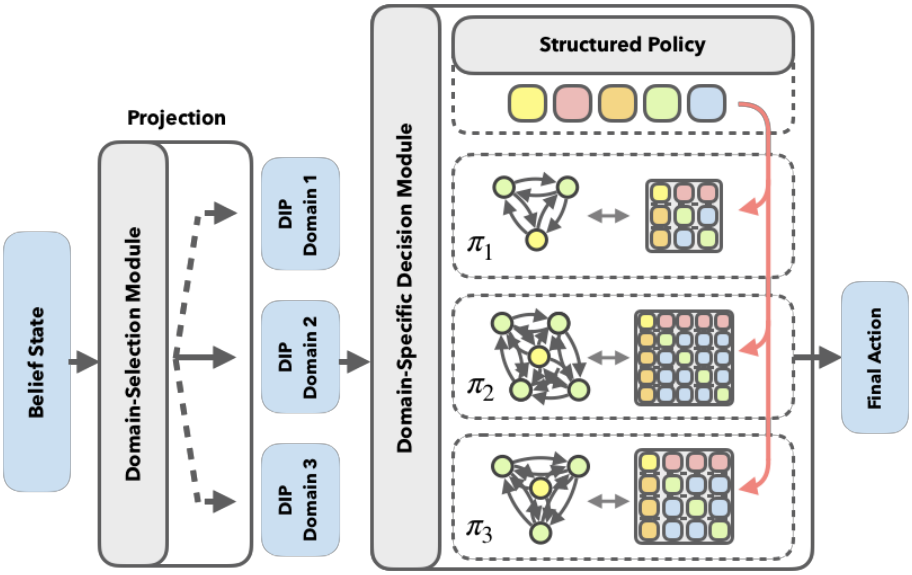


Figure 4.11: Hierarchical and structured policy for multi-domain dialogues with hierarchical decision making at high-level and structured policy with GNN based on weight sharing at low-level.

We build a GNN model composed by three layers in charge to update the nodes representation. First, the GNN transforms inputs (Equation 4.7) Then, it computes the internal nodes representations (Equation 4.8) by following message sending (Equation 4.8b), message aggregation (Equation 4.8c) and representation update (Equation 4.8d). The message sending function $M^l_{i \leftarrow j}$ is a linear transformation with bias. The message aggregation function A^l_i is the average pooling function. The representation update function U^l_i compute the new hidden representation with ReLU activation function and dropout technique during learning stage. Finally, the GNN concatenates all final nodes representations and computes the value or policy functions depending on the final activation function (linear or softmax) (Equation 4.9).

4.4 Experimentation

In this Section, we propose to organise our study by successively analysing the learning efficiency of structured policies with hybrid IL and RL methods in single-domain and multi-domain contexts. On the one hand in Subsection 4.4.1, we propose an evaluation of the dialogue management by establishing an ablation study by progressively adding different levels of structure. In the same time, we analyse the performance of different RL algorithms with different levels of IL.

On the other hand in Subsection 4.4.2, we propose an evaluation of the full dialogue system based on our best approach with simulated and real users to validate structured policies in practice. Notably, we are interested in conducting a more detailed robustness study to raise the limits of the framework used as shown in the previous Chapter 3.

4.4.1 How effective are structured policies for multi-domain dialogues?

We first experimented with PyDial to validate our GNN implementation on single-domain against similar structured methods. Following Casanueva et al. (2017)'s protocol, we tested our agent on different domains and in different environments increasing the inputs' noise and enabling or disabling the action masking mechanism.

Then we experimented with ConvLab to evaluate our approach on multi-domain dialogues. We performed an ablation study: (i) by progressively building from one of the baselines to our proposed GNNs, and (ii) by guiding the exploration with imitation learning.

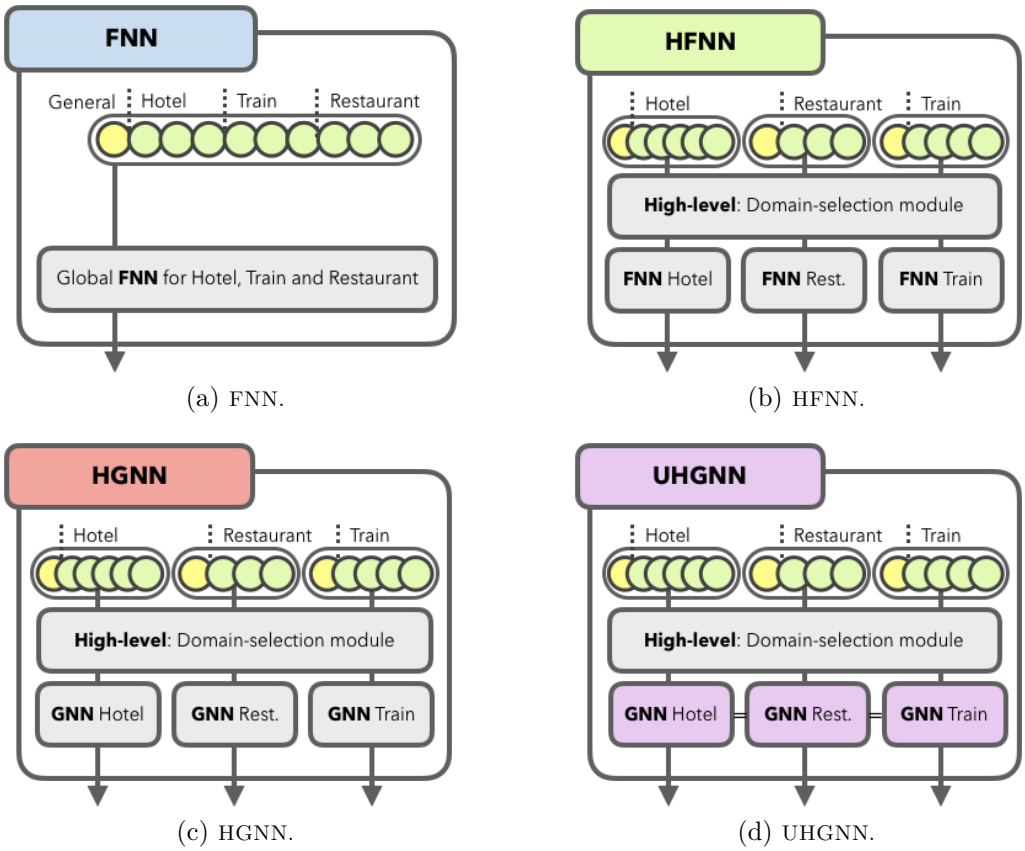
Methods: Table 4.2 shows the compared methods used in PyDial. **DQNGNN** is the DQN-GNN method presented in L. Chen et al. (2018). Structured Actor-Critic (**STRAC**) is its ACER variant also presented in Z. Chen et al. (2020a) that proposes an implicit policy decomposition and uses noisy networks for exploration. **ACGNN** is our implemented ACER-GNN method. Concerning ConvLab2, we have reused the same compared methods that we have already presented in Chapter 3 and that we recall in Table 4.2. All baselines (except for BC and ACER) are from the ConvLab2 framework.

Method name	Abbrev.
Methods for single-domain task-oriented dialogues	
Handcrafted Policy	HDC
Deep Q-learning with Graph Neural Network	DQNGNN
Structured Actor-Critic (based on GNN)	STRAC
Actor-Critic with Graph Neural Network (ours)	ACGNN
Methods for single-domain task-oriented dialogues	
Handcrafted Policy	HDC
Deep Q-Network	DQN
Policy Gradient	PG
Proximal Policy Optimization	PPO
Guided Dialogue Policy Learning	GDPL
Behaviour Cloning	BC
Actor-Critic with Experience Replay	ACER
ACER with Imitation Learning from Oracle Demonstrations	ACER-ILfOD
ACER with Imitation Learning from Oracle Supervision	ACER-ILfOS

Table 4.2: Overview of proposed methods for single and multi-domain task-oriented dialogues.

Ablation study: We propose a wide range of dialogue policies to study the impact of the structure in sample efficiency (as detailed in Table 4.3). An ablation study progressively adds some notion of hierarchy to FNNs to approximate the structure of GNNs. Similarly, we analyse the advantage of sharing a generic GNN among several domains versus specialising a GNN to each domain. Therefore, we propose:

- **Feed-forward Neural Network** (**FNN**) that is a classical feed-forward neural network with DIP parametrisation (Figure 4.12a).
- **Hierarchy of Feed-forward Neural Networks** (**HFNN**) that is a hierarchical policy with hand-crafted domain-selection and FNNs for each domain. Each domain has one corresponding FNN model (Figure 4.12b).
- **Hierarchy of Graph Neural Networks** (**HGNN**) that is a hierarchical policy with hand-crafted domain-selection and GNNs. Each domain has one corresponding GNN model (Figure 4.12c).
- **Hierarchy with Unique Graph Neural Network** (**UHGNN**) that is a HGNN with a unique GNN for all domains. Each domain shares the same GNN model (Figure 4.12d).



(a) FNN. (b) HFNN.

(c) HGNN. (d) UHGNN.

Figure 4.12: Policy and input data structures. Different levels of structure are presented from classical *feed-forward neural network* (FNN) to *graph neural network* (GNN). The prefix H- corresponds to a hierarchical policy and UH- to a unique sub-policy for all domains. For a FNN layer, the input data is the concatenation of all DIP slot representations. For a GNN layer, the input keeps its structure.

Architecture name	Abbrev.
Feed-forward Neural Network (without DIP)	FNN-REF
Feed-forward Neural Network (with DIP)	FNN
Hierarchical Feed-forward Neural Network	HFNN
Hierarchical Graph Neural Network	HGNN
Hierarchical with Unique Graph Neural Network	UHGNN

Table 4.3: Overview of proposed architectures for multi-domain oriented dialogues.

In order to compare with the methods proposed by ConvLab, we have re-implemented **FNN** with DIP and we have named **FNN-REF** the ConvLab's FNN with the native parametrisation (no DIP) and with multiple-actions output. Indeed, ConvLab was built on the annotation convention of MultiWOZ dataset in which the features space was not based on DIP. In particular, the action space is not constrained to the choice of a single action and extends to all composite actions seen in the dataset (*e.g.*, in the search for a restaurant, the agent can ask the user for the expected price and the food at the same time).

When IL is used, the agents learn in simulation and the oracle is the handcrafted policy (both implemented in PyDial or ConvLab). To recall, Imitation Learning from Oracle Demonstrations (ILfOD) is a reinforcement learning method with online data augmentation and Imitation Learning from Oracle Supervision (ILfOS) is a combination of supervised and reinforcement learning with data augmentation.

Hyper-parameters and Training: The FNN models have two hidden layers, both with 128 neurons. The GNN models have one first hidden layer with 64 neurons for each node (two in all: *slot-dependent* node noted S-node and *slot-independent* node noted I-node). Then the second hidden layer is composed of 64 neurons for each relation (three in all: S2S, S2I and I2S). We detail the number of parameters of our models in the Table 4.4. The size of the tested networks are of the order of magnitude of 10 000 to more than 100 000 parameters.

For training stage, we use the Adam optimiser with a learning rate $lr = 0.001$, a dropout rate $dr = 0.1$ and a batch size $bs = 64$. Each loss function has a weight of $\lambda_Q = 0.5$, $\lambda_\pi = 1.$, $\lambda_{IL} = 1.$ and $\lambda_{ent} = 0.01$ respectively. These hyper-parameters have been chosen arbitrarily. The learning frequency is one iteration after each episode (finished dialogue) with only one gradient iteration.

The used oracle is the handcrafted agent proposed by each framework. When we use ILfOD or ILfOS methods, we use in 50% of the time the oracle trajectories. When we use ILfOS, the oracle is constantly called to give us the best expert action as supervision and we use a margin penalty $\mu = \log(2)$.

Our policy algorithm is an off-policy learning that uses experience replay (all data are stored in buffers) without priority (*i.e* without importance sampling). The exploitation-exploration procedure of our implemented models is achieved by Boltz-

Configuration		# of parameters	Configuration		# of parameters
ACER +	FNN REF	87 250	BC +	FNN REF	87 121
	FNN	244 448		FNN	244 319
	HFNN	650 864		HFNN	364 648
	HGNN	375 952		HGNN	192 136
	UHGNN	46 994		UHGNN	24 017

Table 4.4: Number of learned parameters in the models used. We precise that for ACER models, the value and policy functions are learned separately, except for FNN REF and FNN.

mann sampling with a fixed temperature (from $\tau = 1$ to $\tau = 0.1$) depending of the environment. The hyper-parameter search of temperature was performed by grid search.

Experiments: All the experiments were launched 10 times with random initialisations and the results were averaged. Concerning PyDial, we performed two training stages up to 400 and 4 000 dialogues. Concerning ConvLab, each learning trajectory was kept up to 10 000 dialogues with a step of 1 000 dialogues in order to analyse the variability and stability of the methods. In particular, we evaluated each trained agent on 500 dialogues to keep track of the variance.

The system is guided by the rewards as follows. At any time, if all domains are solved (a domain is solved if all related tasks are solved), it gains 20/40 points in PyDial/ConvLab respectively. On the contrary, if the current active domain is solved, it gains 5 points (only in ConvLab). Otherwise, it is penalised by 1 point. The dialogue is marked as successful if and only if both inform recall and book rates are 1. An active domain in dialogue is marked as successful if and only if both inform recall and book rate are 1 in the context of this domain (independently of other domains).

Dialogue Management Evaluation

We evaluate the policy learning algorithms in PyDial and ConvLab for single-domain and multi-domain dialogues respectively.

Single-domain dialogues The results of the evaluation of single-domain dialogues are presented in Table 4.5. We observe that our ACGNN proposal achieves competitive performance in the long training stage in environments with action masking mechanism and with semantic error rates of 0%, 15% and 30% for respectively Env1, Env3 and Env6. ACGNN is better than DQNGNN in a majority of environments. But it seems that our proposal is less efficient when the action masking mechanism is not activated (in Env2 and Env4). ACGNN certainly remains more limited against STRAC in the short training stage and in unfriendly

environment like Env5 since the latter has more advanced learning mechanisms. Our approach manages to reach a performance close to or even better than those of STRAC and those of *handcrafted* (HDC) policies. This confirms that our structured policies are suitable for single-domain task-oriented dialogues.

Task		DQNGNN Suc.	DQNGNN Rew.	STRAC Suc.	STRAC Rew.	ACGNN(Ours) Suc.	ACGNN(Ours) Rew.	HDC Suc.	HDC Rew.
After 400 dialogues									
Env1	CR	85.9	11.0	97.7	13.1	**98.6**	**13.7**	100.0	14.1
0% SER	SFR	52.3	3.4	98.2	**12.3**	**98.3**	**12.3**	97.6	12.1
Mask.	LAP	55.4	4.6	**98.5**	**12.3**	95.7	11.3	97.2	11.8
Env2	CR	75.5	8.1	65.5	5.0	**80.4**	**9.0**	100.0	14.1
0% SER	SFR	71.2	7.1	69.8	4.4	**80.5**	**8.7**	97.6	12.4
No Mask.	LAP	**76.5**	**7.7**	56.9	1.6	69.3	6.1	97.8	11.7
Env3	CR	78.3	8.8	**97.2**	**12.5**	96.2	12.1	95.2	10.8
15% SER	SFR	54.6	3.1	**90.4**	8.9	89.8	**9.2**	90.2	8.9
Mask.	LAP	49.6	2.5	**92.5**	**9.7**	83.5	7.4	88.2	8.4
Env4	CR	70.5	**6.6**	**71.0**	5.2	61.9	5.1	97.0	11.1
15% SER	SFR	66.7	**5.6**	**72.7**	5.0	55.1	4.0	89.2	8.2
No Mask.	LAP	63.7	**4.6**	**65.9**	3.1	44.2	1.1	88.6	8.4
Env5	CR	53.8	2.8	**95.3**	**10.6**	91.3	9.5	94.6	9.2
15% SER	SFR	26.0	-2.9	**80.6**	**4.5**	79.1	**4.5**	87.6	6.3
Mask.	LAP	23.4	-3.0	**87.8**	**6.1**	76.1	2.7	82.8	4.6
Env6	CR	62.8	5.3	**91.9**	**10.3**	91.6	10.1	91.2	9.5
30% SER	SFR	36.2	-0.8	**78.5**	**4.9**	76.3	4.3	80.2	6.5
Mask.	LAP	37.2	-0.3	**84.6**	**6.6**	73.5	3.8	76.6	5.6
After 4000 dialogues									
Env1	CR	74.8	8.7	**99.8**	**14.1**	**99.8**	**14.1**	100.0	14.1
0% SER	SFR	61.3	5.3	98.7	12.7	**99.2**	**12.8**	97.6	12.1
Mask.	LAP	78.5	8.6	97.6	**12.0**	**98.2**	**12.0**	97.2	11.8
Env2	CR	93.6	12.2	**97.9**	**13.1**	87.8	11.9	100.0	14.1
0% SER	SFR	93.0	11.5	**95.6**	**12.1**	91.5	11.8	97.6	12.4
No Mask.	LAP	91.4	11.1	**92.6**	**11.6**	91.8	11.3	97.8	11.7
Env3	CR	96.6	12.6	98.1	**13.0**	**98.2**	12.9	95.2	10.8
15% SER	SFR	89.4	9.4	91.9	**10.5**	**93.8**	10.2	90.2	8.9
Mask.	LAP	84.2	7.9	90.7	**9.7**	**91.8**	9.4	88.2	8.4
Env4	CR	90.9	11.0	**92.9**	**11.5**	87.8	10.9	97.0	11.1
15% SER	SFR	87.7	9.6	**90.2**	**10.7**	70.4	7.4	89.2	8.2
No Mask.	LAP	83.3	8.2	**86.3**	**9.2**	71.7	8.3	88.6	8.4
Env5	CR	95.2	11.2	**97.1**	**11.8**	96.8	11.3	94.6	9.2
15% SER	SFR	82.3	5.9	**89.6**	**8.4**	88.8	7.2	87.6	6.3
Mask.	LAP	70.0	2.8	**88.2**	**6.9**	86.8	5.3	82.8	4.6
Env6	CR	89.3	10.0	92.5	**11.0**	**93.6**	10.9	91.2	9.5
30% SER	SFR	70.8	4.3	81.6	7.0	**84.7**	**7.1**	80.2	6.5
Mask.	LAP	68.7	4.0	83.3	**6.7**	**83.5**	6.4	76.6	5.6

Table 4.5: Average of rewards and success rates on single-domain dialogues with PyDial with 10 different initializations. CR refers to Cambridge Restaurants, SFR to San Francisco Restaurants and LAP to Laptops. SER stands for semantic error rate and Mask refers to action masking mechanism. Env5 is an unfriendly environment (Casanueva et al., 2017). Results of DQNGNN, STRAC and HDC were taken from Z. Chen et al. (2020a). Results in blue represent those with the best rewards and in bold those with the best success rate (excluding HDC).

Multi-domain dialogues Concerning the re-evaluation of the proposed policies in ConvLab, the results of success rates are presented in Figure 4.13. All baselines have difficulties to learn on a horizon of 10 000 dialogues with a clear stagnation of the success rate between 0% and 20%. This is evidence of the difficulties of learning with sparse rewards on multi-domain dialogues.

Figure 4.13: Distribution of the performance of the baselines on multi-domain dialogues in ConvLab, with 10 random initialisations. The colored area represents the distribution and the middle line represents its median.

Then we perform an ablation study based on ACER. The results of this study are presented in Figure 4.14. All variants of ACER (Figure 4.14a) have difficulties to learn in contrast to BC variants (Figure 4.14b) for which we can see an evolution in performance up to 90%. We can notice that using ConvLab's native parameterisation with multiple-actions (FNN REF in blue) is better than using our DIP representations with some macro-actions (FNN in orange). But the successive use of hierarchical decision making (HFNN in green), graph neural network (HGNN in purple) and generic policy (UHGNN in red) drastically improves the performance. And this analysis is still valid when using IL like ILfOD (Figure 4.14c) and ILfOS (Figure 4.14d). Regarding ILfOD, we can observe that neither hierarchical decision making nor separated GNNs guarantee learning stability. On the other hand, learning a generic GNN with weight sharing allows collaborative gradient updating and thus efficient learning on multi-domain dialogues when domain transfer is possible. This suggests that GNN structured networks are useful for learning dialogue policies on multi-domain dialogues and that these policies can be transferred during learning across domains to improve performance.

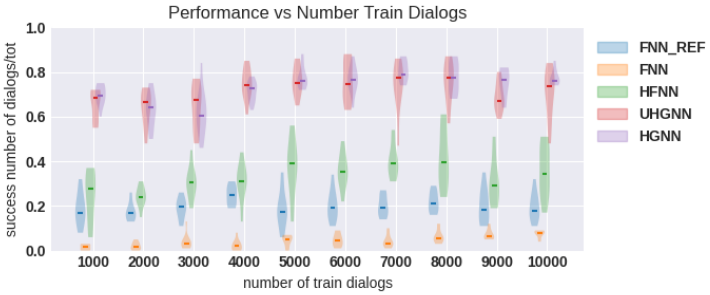
4.4.2 How effective are structured policies within a full system?

We propose to evaluate our best DM policy in a dialogue system with simulated and real users. From now on, we decide to use ACER-ILfOS-UHGNN under a shorter name **ACGOS**.

(a) *ACER* methods

(b) *BC* algorithms.

(c) *ACER* with *ILfOD*.



(d) *ACER* with *ILfOS*.

Figure 4.14: Distribution of the performance of the proposed approaches with different architectures on multi-domain dialogues in ConvLab, with 10 different initialisations. The colored area represents the distribution and the middle line represents its median. *BC* stands for behaviour cloning. *ILfOD* and *ILfOS* stand for imitation learning from oracle demonstrations and from oracle supervision respectively. '*H*' refers to hierarchical decision-making and '*U*' to unique generic method.

Configuration	Avg Turn (succ/all)	Inform (%) Prec. / Rec. / F1	Book Rate (%)	Complete Rate (%)	Success Rate (%)
Dialogue Management					
HDC	10.6/10.6	87.2 / 98.6 / 90.9	98.6	97.9	97.3
ACGOS (ours)	13.3/13.5	97.2 / 98.5 / 97.4	97.8	98.0	87.6
Dialogue System (MILU NLU + hand-crafted NLG)					
HDC	11.5/12.0	79.5 / 89.3 / 82.0	83.9	86.1	75.7
ACGOS (ours)	13.2/16.5	71.3 / 83.5 / 74.3	79.1	74.8	60.4
Dialogue System (BERT NLU + hand-crafted NLG)					
HDC	11.4/12.0	82.8 / 94.1 / 86.2	91.5	92.7	83.8
MLE*	12.2/—	— / — / 67.8	—	46.7	41.9
PG*	12.5/—	— / — / 67.1	—	47.6	42.0
GDPL*	12.2/—	— / — / 67.9	—	49.9	44.0
PPO*	11.6/—	— / — / 71.1	—	64.9	63.8
ACGOS (ours)	14.0/14.8	76.2 / 92.2 / 80.8	87.0	88.5	74.8

Table 4.6: Dialogue system evaluation with simulated users and different configurations. Configurations without NLU and NLG modules represent perfect transmission of the dialogue acts. Configurations with ACGOS and HDC policies are been evaluated on a single run with 1 000 dialogues. Configurations with * are taken from Rohmatillah et al. (2021) and used behaviour cloning as the pre-trained weights. Some missing information has been noted as —.

Simulated-User Evaluation We train it on a horizon of 100 000 dialogues. We compare it with two benchmarks taken from J. Li et al. (2020). The first one consists of a multi-intent language understanding model (MILU), a handcrafted DST, a handcrafted policy, and a template-based NLG. The second one consists of a similar pipeline with a BERT as NLU. We add more baselines from Rohmatillah et al. (2021) that re-evaluate standard policies that use behavior cloning as the pre-trained weights. The results of our experimentation are presented in Table 4.6.

We observe that the performance of our approach is within 10 points near to the handcrafted policy when using perfect NLU and NLG (97.3 *vs.* 87.6) and BERT and template-based NLG (83.8 *vs.* 74.8). It is indeed better compared to the baselines with more than 10 points difference (*e.g.* with 74.8 for ACGOS *vs.* 63.8 for PPO). It is worth noting that compared to other methods ACGOS can reach a performance closer to handcrafted. These results highlight the clear benefit of structured policies against standard policies.

Human Evaluation We evaluate our best policy within a dialogue system with real users. We have organised human evaluation sessions in our research team in which we propose to discuss with our ACGOS dialogue manager with two different NLU modules: MILU and BERT. For the sake of comparability, we show the human evaluation of J. Li et al. (2020), in which a handcrafted policy was used. They reported results on 100 dialogues while we report results on around 45 dialogues. The results of this experimentation are presented in Table 4.7.

Configuration	Avg Turn	Success Rate (%)	Nb of Dial.
BERT + HDC (J. Li et al., 2020)	15.48	65.81 ± 9.48	100
MILU + HDC (J. Li et al., 2020)	17.54	56.45 ± 9.92	100
BERT + ACGOS (Ours)	17.12	57.78 ± 14.73	45
MILU + ACGOS (Ours)	21.36	50.06 ± 14.58	47

Table 4.7: Dialogue system evaluation with real users and different configurations and a 95% confidence level for success rates.

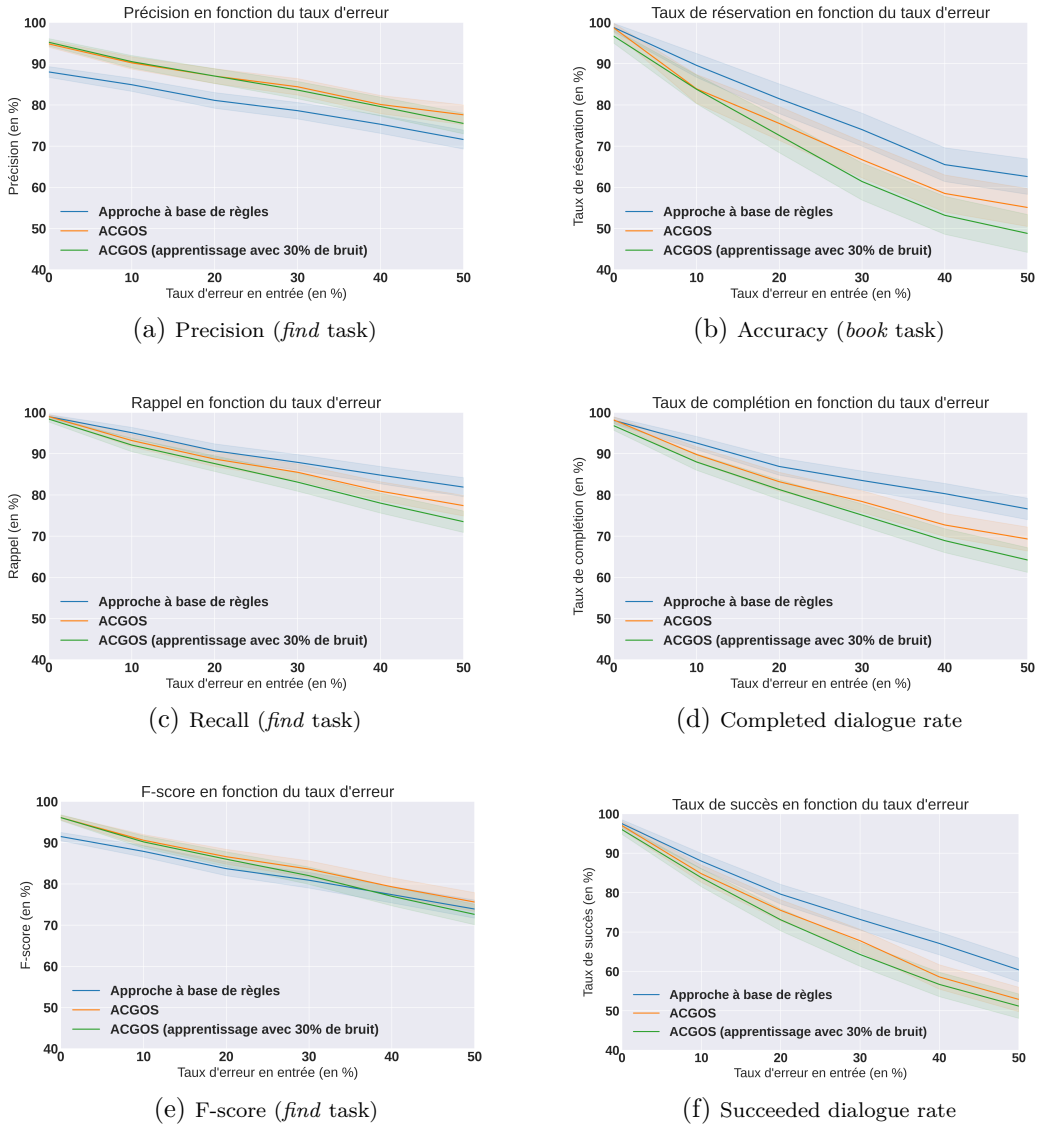
Although real dialogue systems are more complex than simulated environments, the proposed policies are able to succeed more than 50% of the time. Moreover, the performance of our approach is still within 10 points of that of the handcrafted policy. We observe also that using BERT as NLU clearly boost the performance compared to MILU.

4.4.3 How robust are dialogue systems to noisy inputs?

To study the robustness of the DM to noisy inputs, we simulate input noise by imposing a certain error rate on the dialogue acts that the system receives. We can interpret this confusion as emanating from poor speech recognition or poor natural language understanding. In practice, a confusion rate of 20% will lead to a modification of one of the elements of the dialogue act among the predicate, the slot and the value in 20% of the dialogue turns. These modifications are constrained to respect the ontologies imposed by the domains (*i.e.* errors such as INFORM(price=italian)) are forbidden).

In this context, we evaluate our best approach ACGOS in a dialogue system with simulated users and compare it to the handcrafted approach. We vary the confusion rate from 0% to 50% in 10 steps. Evaluations are performed on a set of 1,000 simulated dialogues. The approach is also trained in an environment subjected to a confusion rate of 30% to study the robustness of the approaches and their adaptability in a noisy environment. The results are presented in Figure 4.15.

First of all, we note that the degradation of performance in the face of noise can be explained by the inability of the approaches, when one of the communicated information is corrupted, to propose an offer (task *find*, see Figure 4.15c) but above all by the inability to make a reservation (task *book*, see Figure 4.15b). However, the rule-based approach consistently dominates the structured approaches in terms of success rate (Figure 4.15f). Nevertheless, we observe that the approaches are more cautious (or accurate) when communicating their information since their accuracy remains better (Figure 4.15a). Nevertheless, the tested approaches have a low resistance to noise due to their inability to detect corrupted information even though they have faced noisy inputs during their training. This highlights the need for dialogue systems to develop a specific competence to overcome the lack of robustness to input noise.



(a) Precision (*find* task)

(b) Accuracy (*book* task)

(c) Recall (*find* task)

(d) Completed dialogue rate

(e) F-score (*find* task)

(f) Succeeded dialogue rate

Figure 4.15: Evaluation of the robustness of DM to noisy inputs at a confidence level of 97.5%. Each configuration was evaluated on 1 000 dialogues.

4.5 Conclusion

We studied how structured policies like GNN combined with some imitation learning could be effective to handle truly multi-domain dialogues. The results of our large-scale experiments on ConvLab confirm that an actor-critic with a GNN policy is relevant for solving multi-domain multi-task dialogue problems. Moreover, the results of our experiments on PyDial show that it is relevant for solving the single-domain problem as well. Furthermore, it follows that imitation learning is efficient in improving exploration in multi-domains dialogues. Finally, we evaluated the best policy (ACGOS) in a complete dialogue system with both simulated and real users. The performance of our ACGOS policy overcomes the baselines and is almost on par with the handcrafted policy.

The GNN structured policies combined with imitation learning avoid sparsity, while being data efficient, stable and adaptable. We conclude that a dialogue manager policy structured with GNN is relevant for covering multi-domain multi-task dialogue problems.

Limitations

A limitation of our approach is its dependence on the NLU noisy predictions and on the DST module that provides the active domain. As future work we could integrate the domain selection to our policy as in Z. Chen et al. (2020b), or with a feudal-like approach as in Casanueva et al. (2018a) and Casanueva et al. (2018b). It could be also interesting to study the impact of incorporating real human feed-backs and demonstrations in place of handcrafted teacher.

Nevertheless, robustness to noisy inputs is no longer specifically addressed and our experiments show the need to address this in order to strengthen speech interaction systems. We plan to re-introduce in the context of structured neural approaches already proven solutions, such as a probabilistic dialogue act representation (as proposed in PyDial).

5

Learning Dialogue Policies Faster with Graph Neural Network

Reinforcement Learning has been widely adopted to model DM in task-oriented dialogues. However, the simulators provided with dialogue frameworks are only rough approximations of human behaviour. The ability to learn from a small number of human interactions is hence crucial, especially on multi-domain environments where the action space is large. We therefore propose to use structured policies to improve sample efficiency when learning on these kinds of environments. To this end, we investigate the impact of the policy structure on the dialogue success rate when trained from a limited number of examples. Among the different levels of structure that we tested, Graph Neural Networks (GNNs) show a remarkable superiority by reaching a success rate above 80% with only 50 simulated dialogues. We also investigated the impact of the nature of the provided examples. We found out that learning from simulator examples is easier than learning from human examples. because of the large variability of human strategies, which are less predictable than simulated ones. We therefore suggest to concentrate future research efforts on bridging the gap between human data, simulators and automatic evaluators in dialogue frameworks.

5.1 Motivation

Multi-domain multi-task dialogue systems are designed to complete specific *tasks* in distinct *domains* such as finding and booking a hotel or a restaurant (Zhu et al., 2020). However, the design of a DM is costly: *hand-crafted* policies require a lot of engineer time, pure *supervised learning* requires a lot of expert demonstrations, and pure *reinforcement learning* requires a lot of user interactions to converge. The simulators provided with dialogue frameworks like PyDial (Ultes et al., 2017a) or ConvLab (Zhu et al., 2020) are only rough approximations of the human behaviours and the ability to learn from a small number of human interactions remains crucial. This is especially true on multi-domain and multi-task environments where the action space is large (J. Gao et al., 2018).

A first approach to reduce these costs is to combine expert supervision and reinforcement learning. This approach, called *imitation learning* combined with refining *reinforcement learning*, has been well studied (Hussein et al., 2017) and it offers a balance between expert's and user's costs. But it is not sufficient alone to handle the complexity of multi-domain and multi-task environments.

Another way to reduce these costs is to wire some hand-crafted knowledge about the problem into the policy model, which allows for *few shot learning* (Y. Wang et al., 2020). In particular, structured policies like *graph neural networks* (GNNs) are known to be well suited to handle a variable number of slots and a variable number of domains for the information-seeking task (L. Chen et al. 2018; Z. Chen et al. 2020a).

As an entry point for studying sample efficiency of structured policies, we study the dialogue success rate of these policies once trained in a supervised way from expert demonstrations. We consider two types of demonstrations: *human experts* demonstrations extracted from the MultiWOZ dataset (Pawel Budzianowski et al., 2018), and *simulated experts* demonstrations generated by letting the ConvLab's *hand-crafted* policy interacts with its automatic evaluator.

Our main contribution is to provide a large scale experiment where we study the impact of different levels of structure on policy success rate after a limited number of dialogue demonstrations. For each level of structure, we also compare two sources of demonstrations: simulated dialogues and human dialogues. We show a notable result: our structured policies are able to reach a success rate above 80% with only 50 simulated dialogues in ConvLab.

Another important finding is that in few-shot learning regime, it is harder to reach a high success rate from humans demonstrations than from simulated dialogues. This can be explained first by the large variability of human strategies that is not covered by simulated dialogues which stick to more repetitive – easy to learn – dialogue patterns. Another explanation could be an evaluation bias, simulated dialogues being more in line with artificial evaluators.

The remainder of this Chapter is structured as follows. We present the related work in Section 5.2. Section 5.3 presents the proposed GNNs with IL. The experiments and evaluation are described in Section 5.4. Finally, we conclude in Section 5.5.

5.2 Related Work

Few shot learning takes advantage of prior knowledge to avoid overloading the empirical risk minimiser when the number of available examples is small. In particular, prior knowledge can be used to constrain hypothesis space (*i.e.* model parameters) with parameter sharing or tying in order to reduce reliance on data acquisition and on data annotation (Y. Wang et al., 2020).

Prior knowledge can be built into dialogue systems by imposing a structure in the neural network architecture. A first approach is to use HRL that divides a main problem into several simpler sub-problems. We refer to R. S. Sutton et al. (1999) that introduces Semi-Markov Decision Process using temporal abstraction and to Z. Wen et al. (2020) that introduces Sub-Markov Decision Process using state partition. In the scope of the work (and as detailed in Chapter 4), a *hierarchical policy* corresponds to a meta-controller that chooses to activate a domain and we have one sub-policy per domain (Paweł Budzianowski et al., 2017; Casanueva et al., 2018a; Le et al., 2018a).

In the same vein, Graph Neural Networks (GNNs) have been explored in a wide range of domains because of their empirical success and their theoretical properties which explains its efficiency: the abilities of generalisation, stability and expressiveness (Garcia et al., 2018). GNNs are suitable for applications where the data have a graph structure *i.e.* where the graph outputs are supposed to be node-order equivariant or permutation invariant (Zhou et al., 2020; Wu et al., 2020).

In single-domain dialogue environments, this architecture has been also adapted to model the DM in L. Chen et al. (2018) and Z. Chen et al. (2020a). They have shown that GNNs generalise between similar dialogue slots, manage a variable number of slots and transfer to different domains that perform similar tasks. We thus adopt in this work the DIP (Zhuoran Wang et al., 2015), which standardises the slots representation into a common feature space.

In the continuity of the Chapter 4, we propose to improve multi-domain covering by learning with demonstrations a generic policy based on GNN and with a limited access of expert demonstrations.

5.3 Accelerating Learning from Demonstrations with Structured Policies

In order to investigate the impact of structured policies with Behaviour Cloning in improving sample efficiency in multi-domain multi-task dialogue environments, we briefly introduce the dialogue state and action spaces for structured policies and we present also the different policies and the experts' nature we investigated.

5.3.1 Data structure: Domain Independent Parametrisation

We adopt here the multi-task setting as presented in ConvLab (Zhu et al., 2020), in which a single dialogue can have the following tasks: (i) *find*, in which the system requests information in order to query a database and make an offer; (ii) *book*, in which the system requests information in order to book the item. We adopt the DIP state and action representations, which are not reduced to a flat vector but to a set of sub-vectors. More details can be found in the previous Chapter 4.

For any active domain, the input to the *slot-independent representation* is the concatenation of the previous *slot-independent* user and system actions (see examples of the output below), the number of entities fulfilling the user's constraints in the database, the booleans indicating if the dialogue is terminated and whether an offer has been found / booked. The output corresponds to action scores such as REQMORE, OFFER, BOOK, GREAT, etc.

Regarding the *slot-dependent representation*, its input is composed of the previous *slot-dependent* user and system actions (see output below), the booleans indicating if a value is known and whether the slot is needed for the *find* / *book* tasks. Its output are actions scores such as INFORM, REQUEST and SELECT.

5.3.2 Architecture: Graph Neural Network

Prior knowledge can be integrated in our models by constraining the layer structure imposing symmetries in the neural dialogue policies. Without prior knowledge, the standard structure used is the Feed-forward Neural Network layer (FNN). This unconstrained structure does not assume any symmetry in the network.

Assuming that sub-policies associated with the slots are the same, a better alternative is to use the Graph Neural Network layer (GNN). This structure assumes that the state and action representations have a graph structure that are identically parameterised by DIP. The GNN structure is a fully connected and directed graph, in which each *node* represents a sub-policy associated with a slot and a directed *edge* between two sub-policies represents a message passing. We identify two roles for sub-policies: the general node as I-NODE associated to the *slot-independent representation* and the slot nodes denoted as S-NODE associated to the *slot-dependent representations*. We also identify the relations: I2S for I-NODE to S-NODE, S2I and S2S respectively. More detailed can be found in the previous Chapter 4.

We formally define the GNN structure as follows. Let n be the number of slots and L the number of layers. Let be x the dialogue state, $\mathbf{x}_0 = \phi_0(x)$, $\mathbf{h}_0^l \, \forall l \in [0,L-1]$ and $\mathbf{y}_0$ be respectively the input, hidden and output I-NODE representations. Let the input, hidden and output S-NODES representations be respectively $\forall i \in [1,n]$, $\mathbf{x}_i = \phi_i(x)$, $\mathbf{h}_i^l \, \forall l \in [0,L-1]$ and $\mathbf{y}_i$. First, the GNN transforms inputs:

$$\forall i \in [0,n], \quad \mathbf{h}_i^0 = F_i^0(\phi_i(\mathbf{x})) = \sigma^0(\mathbf{W}_i^0 \phi_i(\mathbf{x}) + \mathbf{b}_i^0) \tag{5.1}$$

Then, at the l-th layer, it computes the hidden nodes representations by following

message sending[1], message aggregation and representation update:

$$\forall i \in [0, n], \quad \mathbf{h}_i^l = F_i^l(\mathbf{h}^{l-1}) \tag{5.2a}$$
$$\mathbf{m}_{i \leftarrow j}^l = M_{i \leftarrow j}^l(\mathbf{h}_j^{l-1}) \tag{5.2b}$$
$$\mathbf{m}_i^l = A_i^l(\mathbf{m}_{i \leftarrow *}^l) \tag{5.2c}$$
$$\mathbf{h}_i^l = U_i^l(\mathbf{m}_i^l) = \sigma^l(\mathbf{m}_i^l) \tag{5.2d}$$

The message sending function $M_{i \leftarrow j}^l$ is a linear transformation with bias. The message aggregation function A_i^l is the average pooling function. The representation update function U_i^l compute the new hidden representation with ReLU activation function and dropout technique during learning stage. Finally, the GNN concatenates ($\oplus$ symbol) all final nodes representations and computes the policy function with the Softmax activation function.

$$\mathbf{y} = \sigma^L(\bigoplus_{i=0}^{n} \mathbf{W}_i^L \mathbf{h}_i^{L-1} + \mathbf{b}_i^L) \tag{5.3}$$

5.3.3 The Structured Policies

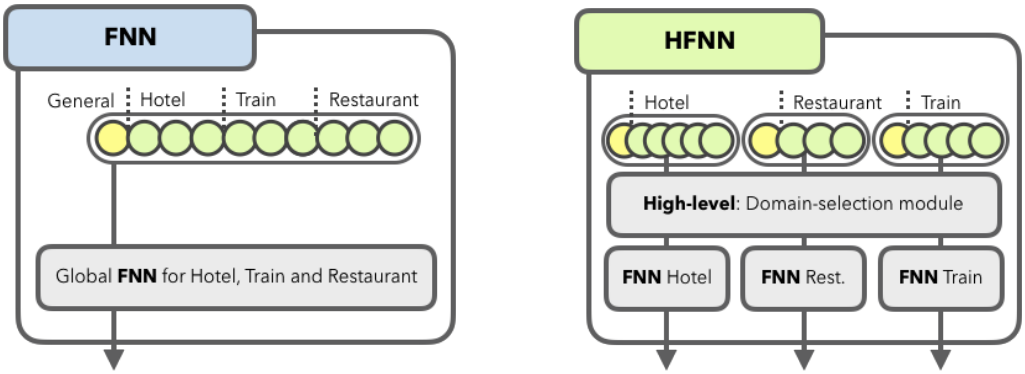
We propose a wide range of dialogue policies to study the impact of the structure in sample efficiency. An ablation study progressively adds some notion of hierarchy to FNNs to approximate the structure of GNNs. Similarly, we analyse the advantage of sharing a generic GNN among several domains versus specialising a GNN to each domain. Therefore, we propose from the least to the most constrained:

- **Feed-forward Neural Network** (**FNN**) that is a classical feed-forward neural network with DIP parametrisation (Figure 5.1a).
- **Hierarchy of Feed-forward Neural Networks** (**HFNN**) that is a hierarchical policy with hand-crafted domain-selection and FNNs for each domain. Each domain has one corresponding FNN model (Figure 5.1b).
- **Hierarchy of Graph Neural Networks** (**HGNN**) that is a hierarchical policy with hand-crafted domain-selection and GNNs. Each domain has one corresponding GNN model (Figure 5.1c).
- **Hierarchy with Unique Graph Neural Network** (**UHGNN**) that is a HGNN with a unique GNN for all domains. Each domain shares the same GNN model (Figure 5.1d).

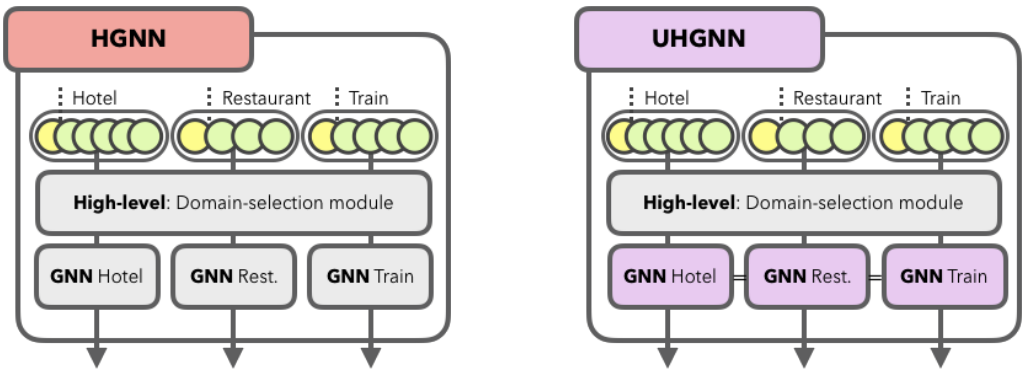
5.3.4 The Expert's Nature

Since our goal is to learn on observed demonstrations delivered by an expert, we propose to focus on policies that learn from both simulated and human experts. For

[1]The notation $i \leftarrow j$ denotes a message sending from slot j to slot i. It corresponds to the directed relation between the slots j and i. The notation $i \leftarrow *$ denotes all messages sending to slot i.



(a) FNN. (b) HFNN.



(c) HGNN. (d) UHGNN.

Figure 5.1: Policy and input data structures. Different levels of structure are presented from classical *feed-forward neural network* (FNN) to *graph neural network* (GNN). The prefix H- corresponds to a hierarchical policy and UH- corresponds to a unique sub-policy for all domains. For a FNN layer, the input data is the concatenation of all DIP slot representations. For a GNN layer, the input keeps its structure.

this purpose, we use the dataset MultiWOZ (Pawel Budzianowski et al., 2018) to follow human experts and the hand-crafted policy of ConvLab (Zhu et al., 2020) as the simulated expert.

Human expert The MultiWOZ dataset is a large annotated and open-sourced collection of human-human chats that covers multiple domains and tasks. Nearly 10k dialogues have been collected by a *Wizard-of-Oz* set-up at relatively low cost and with a small time effort. However, different versions of this dataset corrected and improved the annotations (Eric et al., 2020; Zang et al., 2020; Han et al., 2021; Ye et al., 2021). In this work, we use the MultiWOZ dataset integrated in ConvLab with extended user dialogue act annotations.

Simulated expert The ConvLab framework has been proposed to automatically build, train and evaluate multi-domain multi-task oriented dialogue systems

based on MultiWOZ features. It implements both hand-crafted simulated user and policy. The latter has been shown to be nearly the optimal policy according to the ConvLab evaluation setup of Takanobu et al. (2020). Therefore its use as the simulated expert.

5.4 Experiments

In this section the experimental setup, the proposed models and the evaluation metrics are reviewed.

Experiment Setup: We performed an ablation study by gradually adding different levels of structure from a baseline FNN to the proposed GNN. On the one hand, we analyse the learning efficiency of our models in small training steps. On the other hand, we compare their generalisation ability in few shot learning.

To analyse the learning efficiency, we measure performance with respect to the number of gradient descent steps up to 1 000 iterations with a step size of 100 iterations. We compare learning curves based on randomly chosen 10, 100 and 1 000 training dialogues[2]. We also measure performance as a function of the number of training dialogues available (randomly chosen) namely 10, 50, 100, 500 and 1000 when each training is performed up to 10 000 gradient descent steps. All the experiments were run on ConvLab, restarted 10 times with random initialisation and the results estimated on 500 new dialogues.

Models: The FNN models have two hidden layers, both with 128 neurons. The GNN models have one first hidden layer with 64 neurons for both nodes (S-node and I-node). Then the second hidden layer is composed of 64 neurons for each relation (S2S, S2I and I2S). For training stage, we use the Adam optimiser with a learning rate $lr = 0.001$, a dropout rate $dr = 0.1$ and a batch size $bs = 64$.

Metrics: We evaluate the performance of the policies for all tasks as in ConvLab. Precision, recall and F-score, namely the **inform rates**, are used for the *find* task. Inform recall evaluates whether all the requested information has been informed while inform precision evaluates whether only the requested information has been informed. For the *book* task, the accuracy, namely the **book rate**, is used. It assesses whether the offered entity meets all the constraints specified in the user goal. The dialogue is marked as **successful** if and only if both inform recall and book rate are equal to 1. The dialogue is considered **completed** if it is successful from the user's point of view[3].

[2]These values were chosen arbitrarily to give us an insight into the impact of the number of dialogues on the performance.

[3]A dialog can be completed without being successful if the information provided is not the one objectively expected by the simulator.

Results: First in Subsection 5.4.1, we evaluate the dialogue manager performance when talking to a simulated user. Second in Subsection 5.4.2, we evaluate the learned policies within the entire dialogue system both with simulated and with real users. The evaluations have been done within ConvLab.

5.4.1 Dialogue Manager Evaluation

We analyse our models on the learning efficiency in small training steps and on the ability to generalise in a few-shot setting.

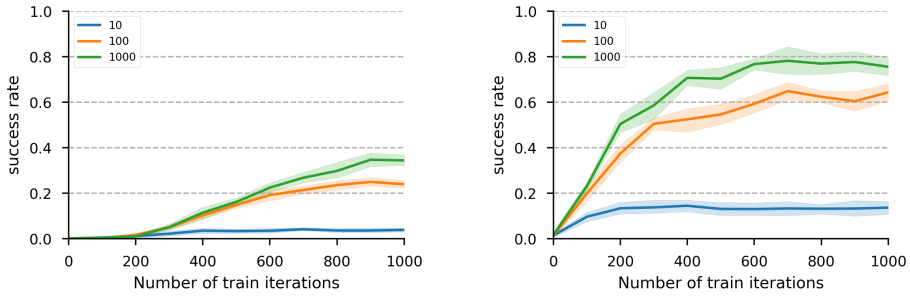
Learning Efficiency Evaluation

We report in Figure 5.2 the results of the ablation study showing the ability of the models to succeed in a short training stage. First, when learning from simulated demonstrations we notice in Figure 5.2a that the baseline (FNN) needs a large number of training dialogues (more than 100) to achieve a moderate performance (less than 40%). We show then in Figure 5.2b that hierarchical networks (HFNN) do improve learning efficiency up to 60% with 100 dialogues, up to 80% with 1 000 dialogues. Finally we show that graph neural network (HGNN in Figure 5.2c) and generic policy (UHGNN in Figure 5.2d) drastically improve the efficiency with few dialogues, more than 60% with 10 dialogues, and achieve remarkable performance above 80% with only 100 dialogues in 1 000 training steps. These observations confirm that hierarchical and generic GNNs allow efficient learning and collaborative gradient update in a short training stage.

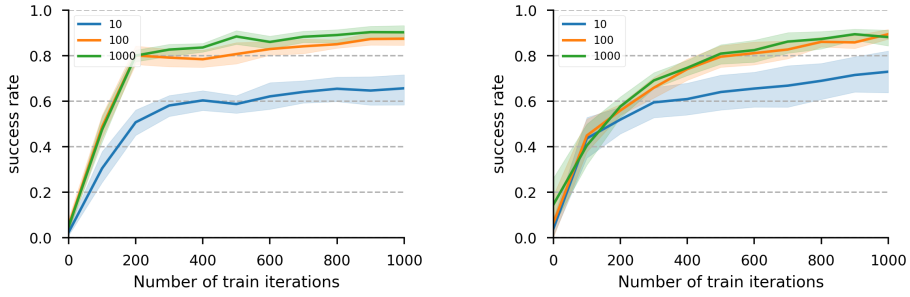
Although standard or hierarchical policies (FNN in Figure 5.2e and HFNN in Figure 5.2f) are less efficient when learning from human demonstrations, they are still above baselines. It is worth noting that structured or generic GNN policies HGNN in Figure 5.2g and UHGNN in Figure 5.2h are able to reach more than 50% success rate.

Few-Shot Learning Evaluation

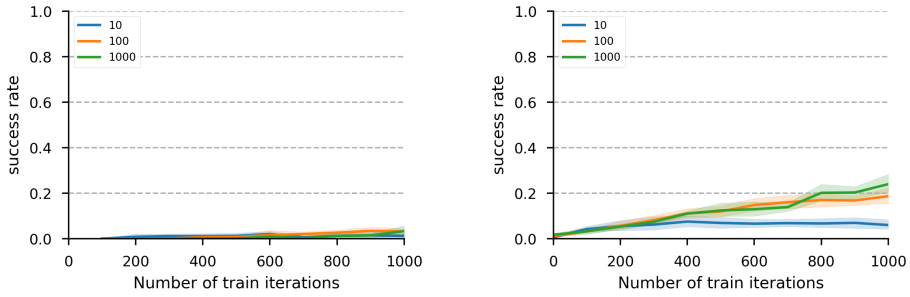
We extended the ablation study in a few-shot scenario focusing on the ability of the models to succeed on specific dialogue tasks as reported in Figure 5.3. In particular, we show the success rate in Figure 5.3a, the inform rate (recall) in Figure 5.3c and the book rate in Figure 5.3e when using simulated demonstrations and respectively in Figure 5.3b, Figure 5.3d and Figure 5.3f when using human demonstrations. The more structured the model, the greater the learning efficiency and the greater the data efficiency. Likewise, we notice that learning is more data-intensive when imitating human strategies. It appears that the booking task is more difficult to perform according to human demonstrations (when comparing Figure 5.3e and Figure 5.3f) or using a flat architecture (FNN gets null results). We therefore foresee that more high quality data is needed to learn on human dialogues.



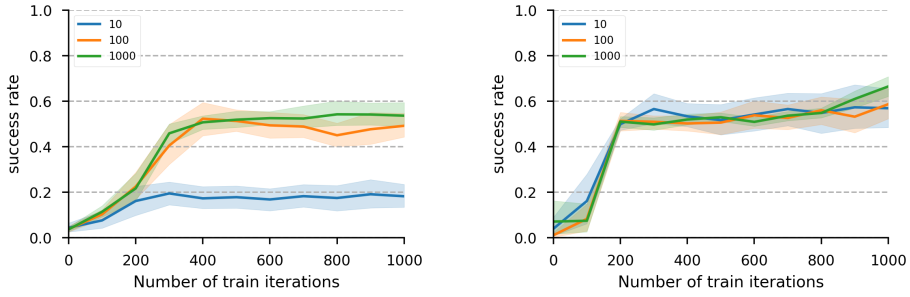
(a) FNN. (b) HFNN.



(c) HGNN. (d) UHGNN.

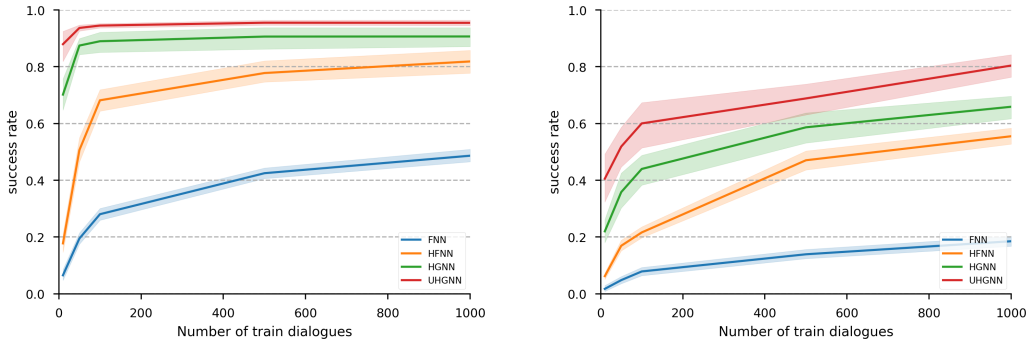


(e) FNN. (f) HFNN.



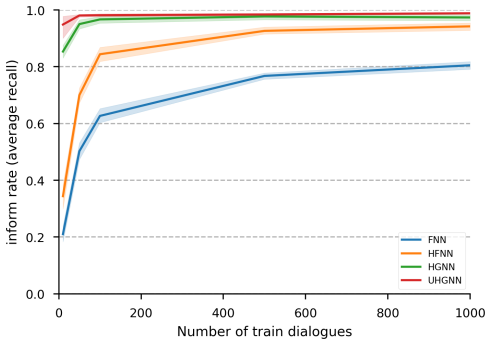
(g) HGNN. (h) UHGNN.

Figure 5.2: Dialogue manager evaluation with simulated users. We present the success rate on 10 / 100 / 1 000 training dialogues as a function of the number of gradient descent steps in a short training scenario. Learning is based on simulated experts ((a) up to (d)) or on human experts ((e) up to (h)). The line plot represents the mean and the coloured area represents the 95% confidence interval over 10 runs.

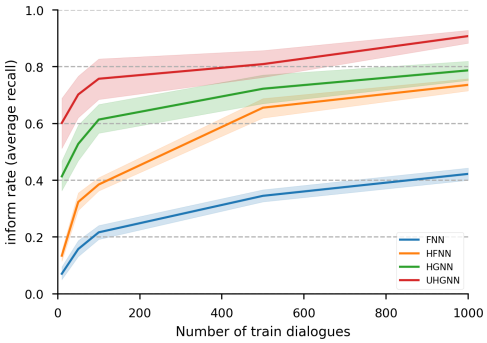


(a) Success rate.

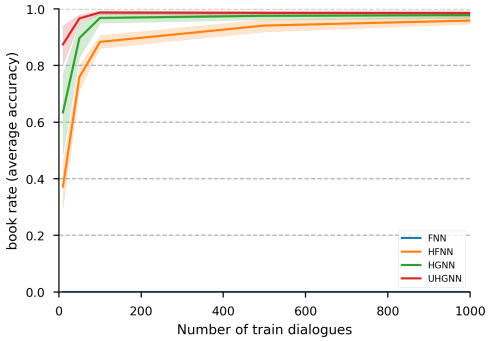
(b) Success rate.



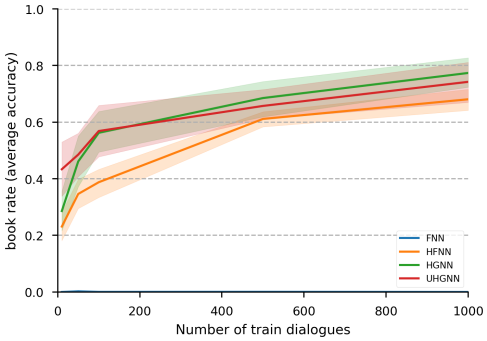
(c) Inform rate (recall).



(d) Inform rate (recall).



(e) Book rate.

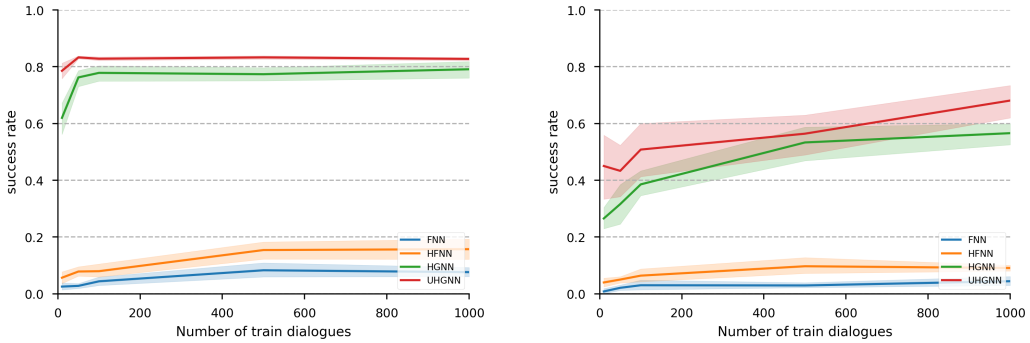


(f) Book rate.

Figure 5.3: Dialogue manager evaluation with simulated user presenting the success rate based on 10 000 training iterations as a function of the number of training dialogues in a long learning scenario. Learning is based on a simulated expert ((a), (c) and (e)) or human experts ((b), (d) and (f)). The line plot represents the mean and the coloured area represents the 95% confidence interval over a sample of 10 runs.

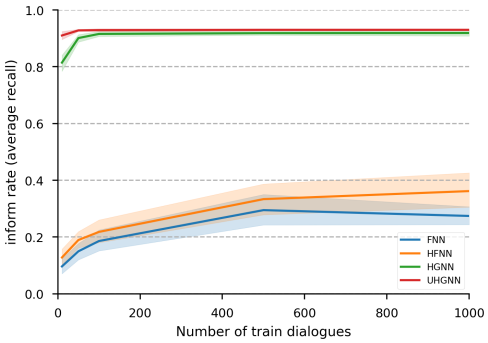
5.4.2 Dialogue System Evaluation

We continue our analysis on the robustness of the studied models with the entire dialogue system facing both simulated and human users. The dialogue system utilises a BERT NLU (Devlin et al., 2019) and a hand-crafted NLG.

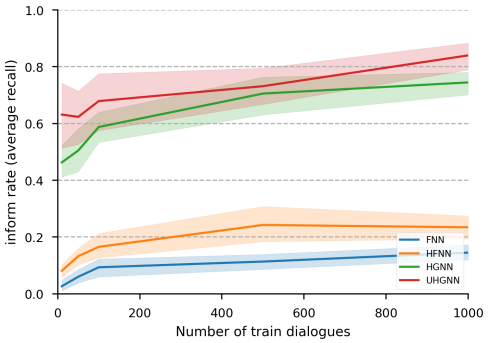


(a) Success rate.

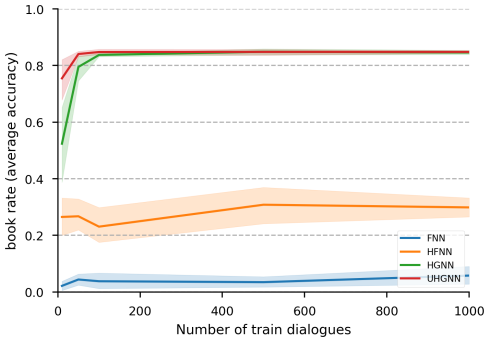
(b) Success rate.



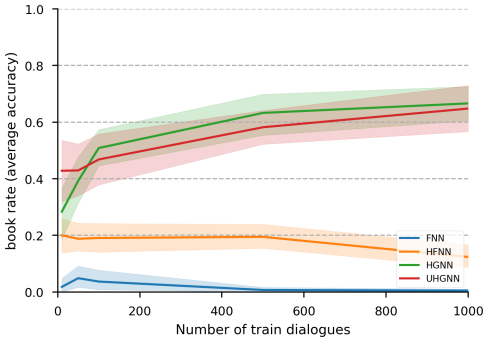
(c) Inform rate (recall).



(d) Inform rate (recall).



(e) Book rate.



(f) Book rate.

Figure 5.4: Dialogue system performance with simulated user based on 10 000 training iterations as a function of the number of training dialogues in a long training scenario. The supervised DM is based on simulated demonstrations ((a), (c), (e)) or on human demonstrations ((b), (d), (f)). The line plot represents the mean and the coloured area represents the 95% confidence interval over a sample of 10 runs.

Simulated User Evaluation

As in the previous subsection, we study the robustness of the models in a few-shot scenario as presented in Figure 5.4. We observe that FNN (in blue) and HFNN (in orange) learning is collapsing when using simulated dialogues (see Figure 5.4a, 5.4c and 5.4e). On the opposite, HGNN (in green) and UHGNN (in red) performance appears more stable in the entire dialogue system even when using real dialogues (see

Configuration	Avg Turn (succ/all)	Inform rate (%) Prec. / Rec. / F1	Book Rate (%)	Complete Rate (%)		Success Rate (%)	
Dialogue Management							
HDC	10.6/10.6	87.2 / 98.6 / 90.9	98.6	97.9	-	97.3	-
MLE-UHGNN-HDC (ours)	12.8/13.0	95.3 / 98.8 / 96.4	98.5	97.3	(-0.6)	95.4	(-1.9)
MLE-UHGNN-MW (ours)	16.5/20.7	94.3 / 90.7 / 91.6	76.7	81.4	(-16.5)	81.0	(-6.3)
Dialogue System (BERT NLU + hand-crafted NLG)							
HDC	11.4/12.0	82.8 / 94.1 / 86.2	91.5	92.7	-	83.8	-
HDC†	11.6/12.3	79.7 / 92.6 / 83.5	91.1	90.5	(-2.2)	81.3	(-2.5)
MLE†	12.1/24.1	62.8 / 69.8 / 62.9	17.6	42.7	(-50.0)	35.9	(-47.9)
PG†	11.0/25.3	57.4 / 63.7 / 56.9	17.4	37.4	(-55.3)	31.7	(-52.1)
GDPL†	11.5/21.3	64.5 / 73.8 / 65.6	20.1	49.4	(-43.3)	38.4	(-45.4)
PPO†	13.1/17.8	69.4 / 85.8 / 74.1	86.6	75.5	(-17.2)	71.7	(-12.1)
MLE-UHGNN-HDC (ours)	14.0/15.4	89.3 / 93.0 / 90.2	84.8	90.0	(-2.7)	82.7	(-1.1)
MLE-UHGNN-MW (ours)	17.0/23.0	84.0 / 87.6 / 84.5	64.8	72.1	(-20.6)	68.1	(-15.7)

Table 5.1: Dialogue manager and system evaluations with simulated users. When evaluating the dialogue manager, the simulated user passes directly dialogue acts and vice-versa. Our tested configurations are evaluated and averaged on 10 run each with 250 dialogues. Configurations with † are taken from the GitHub of ConvLab.

Figure 5.4b, 5.4d and 5.4f). Therefore, these results confirm that behaviour cloning is easier from simulated than human experts. As observed before in Subsection 5.4.2, this can be explained by an large variability of human strategies (hence the need for more data to improve performance). Another explanation is that simulated dialogues are more in line with the artificial evaluator provided in the ConvLab. In addition, it is important not to neglect the side effects of cascading errors due to successive NLU, DST, DM and NLG modules. In particular, the NLU BERT proposed by ConvLab was pre-trained and evaluated on 7 372 user utterances with 14% of errors (F1 86.4%, precision 85.1%, recall 87.8%). This problem can therefore be exacerbated by cascading human errors, as confirmed in the next paragraph.

Finally, we present a detailed comparison table with the best structured policies UHGNN trained on simulated dialogues of ConvLab noted MLE-UHGNN-HDC (HDC for *hand-crafted policy*) and trained on real dialogues of MultiWOZ noted MLE-UHGNN-MW and the baselines of ConvLab (see Table 5.1). In particular, the *maximum likelihood estimator* (MLE) proposed by ConvLab is its implementation of FNN model trained on MultiWOZ corpus in a very long training scenario (multiple passes on all $10k$ dialogues)[4]. Our models show competitive results against ConvLab's baselines, confirming that the structured with supervised learning in few-shot settings is adapted to address the difficulties in multi-task multi-domain dialogues.

Dialogue System (BERT NLU + Rule NLG)	Avg Turn	Satisfaction Rate (%)	Nb of Dial.
HDC	22.6	92.6 ± 9.87	27
MLE-UHGNN-HDC	25.6	50.0 ± 14.8	44
MLE-UHGNN-MW	17.3	36.7 ± 17.2	30

Table 5.2: Dialogue system evaluation with real users with a 95% confidence level for satisfaction rate.

Human Evaluation

We organised preliminary evaluation sessions, in which volunteers were invited to chat on-line with three dialogue systems that were randomly assigned[5]. Subjects do not know which system they are evaluating. Each system has a different DM model: HDC (*hand-crafted policy*), MLE-UHGNN-HDC and MLE-UHGNN-MW combined with the BERT NLU and the hand-crafted NLG provided by ConvLab. At the end of the chat, evaluators were asked whether or not they reach the goal and were satisfied with the performance of the system. The **satisfaction rate** is then the proportion of dialogues in which the system solved the task at the end of the dialogue according to the human evaluator. We reported results on roughly 30 dialogues for each method. The results of this experimentation are presented in Table 5.2. Although test is small-sized and not highly statistically significant, these preliminary results are disconcerting with respect to the simulated ones. The HDC does very well whereas MLE-UHGNN-HDC gets by in half the cases, MLE-UHGNN-MW fails in most cases.

These results can be explained by the limitations of the NLU facing impatient evaluators, short and ambiguous sentences where the active domain is unclear (as in this example of the user saying "What is the name?") or typographical errors. Moreover, it is important to underline that ConvLab does not natively propose the management of uncertainties in the state representation which can strongly restrict the performance of the learning methods in noisy environments.

Another limitation is that the HDC is more adapted to generic dialogues whereas MLE-UHGNNs were trained only on winning dialogues. This implies that learning methods are more sensitive to dialogues that break out of the learned patterns. Similarly, the strategies of simulated and real users do not seem to be well aligned with each other and even more strongly with the expectations of human evaluators.

[4]Another difference is that our models returns one unique action per turn instead of a group of actions.

[5]Crowdsourcing was not used because of ethical concerns regarding the work conditions of collaborators.

5.5 Conclusion

We investigated in this work the impact of structure and experts on success rate in few-shot learning for multi-domain multi-task dialogues. Promising results were obtained: hierarchical and generic GNN policies are able to achieve remarkable performance with few dialogues and few training iterations when following a simulated expert. This confirms the growing interest for these neural structures.

We also present an important finding: the policy performance degrades in few-shot learning when using human demonstrations. This fact questions the alignment between dialogue evaluators and human strategies in state-of-the-art dialogue frameworks.

Limitations

The reduced performance when learning from human experts suggests that we shall concentrate the efforts in bridging the gap between automatic evaluators and high-quality human-human datasets. We also devise the use of *curriculum learning* (Bengio et al., 2009) strategies: starting from simple – simulated – dialogues then adding progressively more complex, human dialogues demonstrations.

It is also necessary to analyse the impact of GNN policies with neural NLU/NLG modules to study how to integrate such structures in end-to-end architectures.

We point out some limitations of ConvLab. The detection of the active domain is sensitive to the output of the NLU and thus sensitive to ambiguous statements. Data representation restricts the DST to a deterministic view and must be adapted to a probabilistic representation to capture the uncertainties in the user's input. Similarly, it may be worthwhile to improve the action space by adding more possibilities for human users, for instance to CONFIRM or DENY in a more flexible way.

Finally, the human evaluation was performed on a small scale and on models trained in a context with few training iterations. A more in-depth or supervised study could shed more light on the raised issues.

III

Conclusion and Perspectives

6

Conclusion

In this thesis, we have explored the use of hierarchical imitation and reinforcement learning techniques for multi-domain task-oriented dialogue systems. Our research focused on addressing the challenges of scalability and efficiency in dialogue policy learning, as well as the need for faster learning and improved performance in multi-domain environments.

6.1 Contributions

Chapter 1 provided a global overview of task-oriented dialogue systems, focusing on the concepts and mechanisms involved in modeling and simulating dialogues. It introduced the fundamental elements necessary to understand the thesis, including the probabilistic framework for dialogue simulation and the essential concepts of task-oriented dialogue systems. The chapter also outlined the approaches and tools used for prototyping dialogue systems, such as data collection and simulation.

Chapter 2 delved into reinforcement learning for dialogue management (DM), highlighted the dynamic and temporal nature of dialogues and the challenges of learning from complex environment and limited interactions. It introduced both reinforcement learning and demonstration-based learning approaches and emphasised the need for scalability and efficiency in dialogue policy learning. It naturally introduced the core concept of the thesis, i.e. hierarchical imitation and reinforcement learning as an effective strategy for adapting policies across different domains or tasks. It discussed the use of expert demonstrations to guide the agent's decision-making process, as well as the use of hierarchical structure in dialogue policy to catalyse the learning for efficient exploration and exploitation.

Chapter 3 explored the integration of expert demonstrations into dialogue policies for improved exploration in task-oriented dialogue systems. It discussed the benefits of hybrid imitation and reinforcement learning strategies, where the agent learns from a sub-optimal expert policy during the learning process. The chapter

presented experiments that demonstrate the effectiveness of this hybrid approach in noisy environments, particularly in single-domain dialogue contexts. It also raised questions about the applicability of these strategies in multi-domain dialogues and suggests avenues for future research, such as exploring different strategies for efficiently incorporating expert demonstrations and feedback.

Chapter 4 focused on structuring dialogue policies using hierarchical reinforcement learning based on graph neural networks (GNNs). It addressed the challenges of handling multi-domain dialogues and complex single-domain dialogues with limited reward signals. The chapter presented structured policies based on GNNs and discussed their effectiveness in multi-domain dialogue problems. It highlighted the benefits of leveraging similarities between domains and slots, as well as the efficiency gains achieved through structured policies. The chapter concluded by emphasizing the relevance of GNN-based policies in multi-domain and single-domain dialogue scenarios.

Chapter 5 investigated the use of structured policies, specifically graph neural networks, to improve sample efficiency in active learning on multi-domain multi-task dialogues. It examined the impact of policy structure and expert guidance on the success rate of few-shot learning. The chapter presented promising results, demonstrating that hierarchical and generic GNN policies can achieve remarkable performance with limited training data. However, it also highlighted the challenges of aligning dialogue evaluators with human strategies and suggested future research directions, such as bridging the gap between automatic evaluators and high-quality human-human datasets and employing curriculum learning strategies. The chapter concluded by emphasizing the need to integrate GNN policies with neural natural language understanding/generation modules and to address the limitations of existing dialogue frameworks.

6.2 Perspectives

Our thesis has shed light on various aspects of task-oriented systems and explored their potential for further development. However, we believe it is crucial to expand the scope of our research to address emerging challenges and advance the field of AI in meaningful ways.

One promising avenue for future investigation is the integration of human feedback into reinforcement learning. By incorporating human preferences as a reward signal, we can enhance the learning process and align AI systems with human values. This approach offers a more nuanced and adaptable framework compared to relying solely on dialogue success as a reward signal.

Moreover, curriculum-based learning offers significant potential to enhance and guide policy dialogue learning. Guiding learning from simple to complex problems in a step-by-step manner can facilitate more efficient and effective knowledge acqui-

sition. This approach enables AI systems to tackle increasingly sophisticated tasks while ensuring a solid foundation of learning.

We also urge researchers to examine the impact of architecture on learning effectiveness to find alternatives to the continuous growth in architecture size. Building upon the success of architectures such as Transformers and GNN-type architectures, we encourage the exploration of alternative architectural designs, leveraging techniques that optimize performance without excessive resource requirements.

Furthermore, while our research has focused on task-oriented systems, the emergence of generative models prompts us to consider their potential for specific tasks. We must investigate how to effectively harness the capabilities of generative models, such as chitchat LLMs, within constrained domains. Ensuring that their usage remains within the intended domain is essential to maintain reliability.

As the field of generative AI continues to advance, ethical considerations become paramount. Researchers must engage in thoughtful discussions surrounding the ethical implications of their results. Balancing the power of generative AI with the need for explanation and interpretability is a central dilemma. Striving for trusted AI that is both powerful and accountable is crucial to ensure responsible deployment and adoption, in the era of regulation of the design and use of AI.

Finally, it is essential to develop robust mechanisms to prevent users from bypassing the controls put in place to guarantee correct and responsible usage of AI systems. Safeguards must be established to mitigate the risk of misuse or unintended consequences. Striking a balance between user flexibility and system control will be a critical aspect of AI system development moving forward.

In conclusion, our thesis marks a step forward in the search for task-oriented systems on the notions of efficiency and scalability, but it also highlights the pressing need for further exploration and development. By addressing the proposed research directions and considering the ethical implications, we can shape the future of AI towards more powerful, accountable, and beneficial systems.

Glossary

A3C Asynchronous Advantage Actor-Critic 53
ACER Actor Critic with Experience Replay 53, 88
AI Artificial Intelligence 1–3, 5, 11, 17
ASR Automatic Speech Recognition 16, 18

BC Behaviour Cloning 43, 65, 76, 78, 79, 100, 113

DIP Domain Independent Parametrisation 88–90, 94, 98, 102, 113, 114
DL Deep Learning 2, 54, 63
DM Dialogue Manager 16–20, 27, 33, 37, 39, 41, 42, 73, 81, 83, 88, 89, 108, 109, 111, 113, 137
DP Dialogue Policy 4, 17
DP Dynamic Programming 50
DQN Deep Q-Network 53, 55, 57, 88
DRL Deep Reinforcement Learning 53, 54
DS Dialogue System 3, 4, 33
DST Dialogue State Tracking 4, 17, 36, 90, 98
DSTC Dialogue State Tracking Challenge 22, 29

FNN Feed-forward Neural Network 89, 92, 94, 96, 114

GNN Graph Neural Network 58, 81, 82, 87–90, 92, 93, 95–100, 105, 109–114, 135, 136
GPI Generalised Policy Iteration 49

HRL Hierarchical Reinforcement Learning 54, 57, 81–85, 88, 113

IL Imitation Learning 42–44, 57, 58, 62, 63, 81, 82, 99, 102, 112
ILfOD Imitation Learning from Oracle Demonstrations 66–69, 72, 76, 78, 100, 102, 134
ILfOF Imitation Learning from Oracle Feed-backs 67, 68, 72, 134
ILfOS Imitation Learning from Oracle Supervision 68, 69, 76, 78, 79, 100, 102, 134
IRL Inverse Reinforcement Learning 42, 44, 45, 62

KB Knowledge Base 16, 17

LfD Learning from Demonstrations 42

MC Monte Carlo 50
MDP Markov Decision Process 33–37, 40, 43, 44, 50, 84, 86
ML Machine Learning 2–4
MLE Maximum Likelihood Estimation 56

NLG Natural Language Generation 4, 16–18, 41
NLP Natural Language Processing 3, 4
NLU Natural Language Understanding 4, 16, 18, 41, 91

POMDP Partial Observable Markov Decision Process 4, 36
PPO Proximal Policy Optimization 53

RL Reinforcement Learning 4, 5, 33, 40, 43, 44, 46–49, 52–58, 61–66, 72, 78, 81, 82, 99, 111, 134

SDS Spoken Dialogue System 19, 33
SER Semantic Error Rate 28
SL Supervised Learning 43, 56, 62
SMDP Semi-Markov Decision Process 83, 113
SubMDP Sub-Markov Decision Process 83–86, 113, 139

TD Temporal Difference 50
TRACER Trust Region Actor Critic with Experience Replay 53
TRPO Trust Region Policy Optimization 53
TTS Text-To-Speech 16–18

WOZ Wizard-of-Oz 25, 26

List of Figures

List of Tables

Bibliography

References

Abbeel, Pieter and Andrew Y. Ng (2004). "Apprenticeship learning via inverse reinforcement learning". en. In: *Twenty-first international conference on Machine learning - ICML '04*. Banff, Alberta, Canada: ACM Press, p. 1. DOI: 10.1145/1015330.1015430. URL: http://portal.acm.org/citation.cfm?doid=1015330.1015430 (visited on 11/14/2019).

Androutsopoulos, Ion and Graeme Ritchie (2000). "Database interfaces". In: *Handbook of Natural Language Processing*, pp. 209–240.

Argall, Brenna D., Sonia Chernova, Manuela Veloso, and Brett Browning (May 2009). "A survey of robot learning from demonstration". en. In: *Robotics and Autonomous Systems* 57.5, pp. 469–483. ISSN: 09218890. DOI: 10.1016/j.robot.2008.10.024. URL: https://linkinghub.elsevier.com/retrieve/pii/S0921889008001772 (visited on 09/28/2020).

Asri, Layla El, Jing He, and Kaheer Suleman (2016). "A sequence-to-sequence model for user simulation in spoken dialogue systems". In: *arXiv preprint arXiv:1607.00070*.

Asri, Layla El, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman (2017). "Frames: a corpus for adding memory to goal-oriented dialogue systems". In: *arXiv preprint arXiv:1704.00057*.

Attia, Alexandre and Sharone Dayan (Jan. 2018). "Global overview of Imitation Learning". en. In: *arXiv:1801.06503 [cs, stat]*. arXiv: 1801.06503. URL: http://arxiv.org/abs/1801.06503 (visited on 05/25/2020).

Austin, John Langshaw (1962). "How to do things with words". In: *Cambridge (Mass.)*, pp. 2005–168.

— (1975). *How to do things with words*. Oxford university press.

Bacon, Pierre-Luc, Jean Harb, and Doina Precup (2016). "The Option-Critic Architecture". In: *AAAI Conference on Artificial Intelligence*.

Bellman, Richard (1957). “A Markovian decision process”. In: *Journal of mathematics and mechanics*, pp. 679–684.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston (2009). “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48.

Bordes, Antoine, Y-Lan Boureau, and Jason Weston (2016). “Learning end-to-end goal-oriented dialog”. In: *arXiv preprint arXiv:1605.07683*.

Boularias, Abdeslam, Hamid R Chinaei, and Brahim Chaib-draa (2010). “Learning the Reward Model of Dialogue POMDPs from Data”. en. In: p. 9.

Brabra, Hayet, Marcos Báez, Boualem Benatallah, Walid Gaaloul, Sara Bouguelia, and Shayan Zamanirad (2021). “Dialogue management in conversational systems: a review of approaches, challenges, and opportunities”. In: *IEEE Transactions on Cognitive and Developmental Systems*.

Budzianowski, Paweł, Stefan Ultes, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gašić (2017). “Sub-domain modelling for dialogue management with hierarchical reinforcement learning”. In: *arXiv preprint arXiv:1706.06210*.

Budzianowski, Pawel, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic (2018). “MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling”. In: *EMNLP*.

Carrara, Nicolas (2019). “Reinforcement learning for Dialogue Systems optimization with user adaptation.” PhD thesis. Ecole Doctoral Science pour l’Ingénieur Université Lille Nord-de-France.

Casanueva, Iñigo, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić (Nov. 2017). “A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management”. en. In: *arXiv:1711.11023 [cs, stat]*. arXiv: 1711.11023. URL: http://arxiv.org/abs/1711.11023 (visited on 12/12/2019).

Casanueva, Iñigo, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M Rojas Barahona, Bo-Hsiang Tseng, and Milica Gasic (2018a). “Feudal Reinforcement Learning for Dialogue Management in Large Domains”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 714–719.

Casanueva, Iñigo, Paweł Budzianowski, Stefan Ultes, Florian Kreyssig, Bo-Hsiang Tseng, Yen-Chen Wu, and Milica Gasic (2018b). “Feudal dialogue management with jointly learned feature extractors”. In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 332–337.

Chandramohan, Senthilkumar, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin (2011a). “User Simulation in Dialogue Systems using Inverse Reinforcement Learning”. en. In: *Proc. Annu. Conf. Int. SpeechCommun. Assoc., Florence, Italy*, pp. 1025–1028. (Visited on 01/08/2020).

— (2011b). “User simulation in dialogue systems using inverse reinforcement learning”. In: *Twelfth annual conference of the international speech communication association*.

— (Sept. 2012). “Behavior Specific User Simulation in Spoken Dialogue Systems”. In: *10th {ITG} Conference on Speech Communication*. Braunschweig, Germany, pp. 1–4. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6309603.

Chang, Kai-Wei, Akshay Krishnamurthy, and Alekh Agarwal (2015). “Learning to Search Better than Your Teacher”. en. In: *International Conference on Machine Learning (ICML)*, p. 9.

Chen, Hongshen, Xiaorui Liu, Dawei Yin, and Jiliang Tang (2017). “A survey on dialogue systems: Recent advances and new frontiers”. In: *Acm Sigkdd Explorations Newsletter* 19.2, pp. 25–35.

Chen, Lu, Bowen Tan, Sishan Long, and Kai Yu (2018). “Structured dialogue policy with graph neural networks”. In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1257–1268.

Chen, Lu, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu (2017). “Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning”. en. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2454–2464. DOI: 10.18653/v1/D17-1260. URL: http://aclweb.org/anthology/D17-1260 (visited on 10/11/2020).

Chen, Zhi, Lu Chen, Xiaoyuan Liu, and Kai Yu (2020a). “Distributed Structured Actor-Critic Reinforcement Learning for Universal Dialogue Management”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing, pp. 2400–2411. ISSN: 2329-9304. DOI: 10.1109/TASLP.2020.3013392.

Chen, Zhi, Xiaoyuan Liu, Lu Chen, and Kai Yu (2020b). “Structured hierarchical dialogue policy with graph neural networks”. In: *arXiv preprint arXiv:2009.10355*.

Clark, Herbert H (1996). *Using language*. Cambridge university press.

Cuayáhuitl, Heriberto (2009). “Hierarchical Reinforcement Learning for Spoken Dialogue Systems”. PhD Thesis. University of Edinburgh. URL: https://www.era.lib.ed.ac.uk/bitstream/handle/1842/2750/Cuayahuitl%20PhD%202009.pdf?sequence=6&isAllowed=y (visited on 01/07/2020).

Cuayáhuitl, Heriberto, Seunghak Yu, Ashley Williamson, and Jacob Carse (2016). “Deep reinforcement learning for multi-domain dialogue systems”. In: *arXiv preprint arXiv:1611.08675*.

Dahlbäck, Nils, Arne Jönsson, and Lars Ahrenberg (1993). “Wizard of Oz studies—why and how”. In: *Knowledge-based systems* 6.4, pp. 258–266.

Daigavane, Ameya, Balaraman Ravindran, and Gaurav Aggarwal (2021). “Understanding Convolutions on Graphs”. In: *Distill*. https://distill.pub/2021/understanding-gnns. DOI: 10.23915/distill.00032.

Daubigney, Lucie, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin (2012). "A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization". In: *IEEE Journal of Selected Topics in Signal Processing* 6, pp. 891–902.

Daumé, Hal, John Langford, and Daniel Marcu (2009). "Search-based structured prediction". In: *Machine learning* 75.3, pp. 297–325.

Dayan, Peter and Geoffrey E Hinton (1993). "Feudal reinforcement learning". In: *Advances in neural information processing systems*, pp. 271–278.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of NAACL-HLT*, pp. 4171–4186.

Dietterich, Thomas G (2000). "Hierarchical reinforcement learning with the MAXQ value function decomposition". In: *Journal of artificial intelligence research* 13, pp. 227–303.

Eckert, Wieland, Esther Levin, and Roberto Pieraccini (1997). "User modeling for spoken dialogue system evaluation". In: *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE, pp. 80–87.

Eksombatchai, Chantat, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec (2018). "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time". In: *Proceedings of the 2018 world wide web conference*, pp. 1775–1784.

Engel, Yaakov, Shie Mannor, and Ron Meir (Aug. 2005). "Reinforcement learning with Gaussian processes". In: DOI: 10.1145/1102351.1102377. (Visited on 12/10/2019).

Eric, Mihail, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür (2020). "MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines". In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France. European Language Resources Association, pp. 422–428.

Eysenbach, Ben, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov (2020). "Rewriting history with inverse rl: Hindsight inference for policy improvement". In: *Advances in neural information processing systems* 33, pp. 14783–14795.

Fatemi, Mehdi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman (Sept. 2016). "Policy Networks with Two-Stage Training for Dialogue Systems". en. In: *arXiv:1606.03152 [cs]*. arXiv: 1606.03152. URL: http://arxiv.org/abs/1606.03152 (visited on 12/12/2019).

Ferreira, Emmanuel and Fabrice Lefèvre (Aug. 2013). "Social signal and user adaptation in reinforcement learning-based dialogue management". In: *2nd Workshop on Machine Learning for Interactive Systems - Bridging the Gap Between Perception, Action and Communication, MLIS-IJCAI 2013*. Beijing, China: ACM Press, pp. 61–69. DOI: 10.1145/2493525.2493535. URL: http://doi.acm.org/10.1145/2493525.2493535 (visited on 01/13/2020).

Ferreira, Emmanuel and Lefèvre, Fabrice (2013). "Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management". In:

Proceedings of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013, pp. 108–113.

Florensa, Carlos, Yan Duan, and P. Abbeel (2016). "Stochastic Neural Networks for Hierarchical Reinforcement Learning". In: *ArXiv* abs/1704.03012.

Fortunato, Meire, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg (2017). "Noisy Networks for Exploration". In: *ArXiv* abs/1706.10295.

Fraser, Norman M and G Nigel Gilbert (1991). "Simulating speech systems". In: *Computer Speech & Language* 5.1, pp. 81–99.

Fu, Justin, Katie Luo, and Sergey Levine (Aug. 2018). "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning". en. In: *arXiv:1710.11248 [cs]*. arXiv: 1710.11248. URL: http://arxiv.org/abs/1710.11248 (visited on 09/24/2020).

Gama, Fernando, Joan Bruna, and Alejandro Ribeiro (2020). "Stability properties of graph neural networks". In: *IEEE Transactions on Signal Processing* 68, pp. 5680–5695.

Gao, Jianfeng, Michel Galley, and Lihong Li (2018). "Neural approaches to conversational ai". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1371–1374.

Gao, Yang, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell (May 2019). "Reinforcement Learning from Imperfect Demonstrations". en. In: *arXiv:1802.05313 [cs, stat]*. arXiv: 1802.05313. URL: http://arxiv.org/abs/1802.05313 (visited on 07/08/2020).

Garcia, Victor and Joan Bruna (2018). "Few-shot learning with graph neural networks". In: *6th International Conference on Learning Representations, ICLR 2018*.

Gasic, Milica, Dongho Kim, Pirros Tsiakoulis, and Steve J. Young (2015). "Distributed dialogue policies for multi-domain statistical dialogue management". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5371–5375.

Gasic, Milica and Steve Young (2013). "Gaussian processes for POMDP-based dialogue manager optimisation". en. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, p. 13. (Visited on 12/10/2019).

Geishauser, Christian, Songbo Hu, Hsien-chin Lin, Nurul Lubis, Michael Heck, Shutong Feng, Carel van Niekerk, and Milica Gašić (2021). "What Does The User Want? Information Gain for Hierarchical Dialogue Policy Optimisation". In: *arXiv preprint arXiv:2109.07129*.

Geist, Matthieu and Olivier Pietquin (2010). "Kalman temporal differences". In: *Journal of artificial intelligence research* 39, pp. 483–532.

Ghavamzadeh, Mohammad, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. (2015). "Bayesian reinforcement learning: A survey". In: *Foundations and Trends® in Machine Learning* 8.5-6, pp. 359–483.

Goecks, Vinicius G., Gregory M. Gremillion, Vernon J. Lawhern, John Valasek, and Nicholas R. Waytowich (Apr. 2020). "Integrating Behavior Cloning and Reinforcement Learning for Improved Performance in Dense and Sparse Reward Environments". en. In: *arXiv:1910.04281 [cs, stat]*. arXiv: 1910.04281. URL: http://arxiv.org/abs/1910.04281 (visited on 09/25/2020).

Gordon-Hall, Gabriel, Philip John Gorinski, and Shay B. Cohen (Aug. 2020). "Learning Dialog Policies from Weak Demonstrations". en. In: *arXiv:2004.11054 [cs]*. arXiv: 2004.11054. URL: http://arxiv.org/abs/2004.11054 (visited on 09/25/2020).

Grice, Herbert P (1975). "Logic and conversation". In: *Speech acts*. Brill, pp. 41–58.

Haarnoja, Tuomas, Haoran Tang, Pieter Abbeel, and Sergey Levine (July 2017). "Reinforcement Learning with Deep Energy-Based Policies". en. In: *arXiv:1702.08165 [cs]*. arXiv: 1702.08165. URL: http://arxiv.org/abs/1702.08165.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (Aug. 2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". en. In: *arXiv:1801.01290 [cs, stat]*. arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30.

Han, Ting, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang (2021). "MultiWOZ 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation". In: *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, pp. 206–218.

Hasselt, Hado van, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil (Dec. 2018). "Deep Reinforcement Learning and the Deadly Triad". en. In: *arXiv:1812.02648 [cs]*. arXiv: 1812.02648. URL: http://arxiv.org/abs/1812.02648 (visited on 09/24/2020).

Hasselt, Hado van, Arthur Guez, and David Silver (Dec. 2015). "Deep Reinforcement Learning with Double Q-learning". en. In: *arXiv:1509.06461 [cs]*. arXiv: 1509.06461. URL: http://arxiv.org/abs/1509.06461 (visited on 09/25/2020).

Hester, Todd, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys (Nov. 2017). "Deep Q-learning from Demonstrations". en. In: *arXiv:1704.03732 [cs]*. arXiv: 1704.03732. URL: http://arxiv.org/abs/1704.03732 (visited on 09/25/2020).

Hester, Todd, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. (2018). "Deep q-learning from demonstrations". In: *Thirty-second AAAI conference on artificial intelligence*.

Ho, Jonathan and Stefano Ermon (2016). "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29.

Houthooft, Rein, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel (Jan. 2017). "VIME: Variational Information Maximizing Exploration". en. In: *arXiv:1605.09674 [cs, stat]*. arXiv: 1605.09674. URL: http://arxiv.org/abs/1605.09674 (visited on 10/05/2020).

Howard, Ronald A (1960). "Dynamic programming and markov processes." In.

Huang, Xinting, Jianzhong Qi, Yu Sun, and Rui Zhang (May 2020). "Semi-Supervised Dialogue Policy Learning via Stochastic Reward Estimation". en. In: *arXiv:2005.04379 [cs]*. arXiv: 2005.04379. URL: http://arxiv.org/abs/2005.04379 (visited on 09/25/2020).

Hussein, Ahmed, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne (2017). "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2, pp. 1–35.

Hussenot, Léonard, Robert Dadashi, Matthieu Geist, and Olivier Pietquin (2021). "Show me the Way: Intrinsic Motivation from Demonstrations". In: *AAMAS 2021-20th International Conference on Autonomous Agents and Multiagent Systems*.

Jefferson, Geil (1972). "Side sequences". In: *Studies in social interaction*.

Jiang, Weiwei and Jiayun Luo (2022). "Graph neural network for traffic forecasting: A survey". In: *Expert Systems with Applications*, p. 117921.

Jospin, Laurent Valentin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun (2022). "Hands-on Bayesian neural networks—A tutorial for deep learning users". In: *IEEE Computational Intelligence Magazine* 17.2, pp. 29–48.

Jurafsky, Dan and James H Martin (2014). "Speech and language processing. Vol. 3". In: *US: Prentice Hall*.

Kelley, John F (1984). "An iterative design methodology for user-friendly natural language office information applications". In: *ACM Transactions on Information Systems (TOIS)* 2.1, pp. 26–41.

Kim, Seokhwan, Michel Galley, Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, et al. (2021). "Overview of the eighth dialog system technology challenge: DSTC8". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29, pp. 2529–2540.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

Kulkarni, Tejas D., Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum (2016). "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation". In: *ArXiv* abs/1604.06057.

Le, Hoang, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé III (2018a). "Hierarchical imitation and reinforcement learning". In: *International conference on machine learning*. PMLR, pp. 2917–2926.

Le, Hoang, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III (Mar. 2018b). "Hierarchical Imitation and Reinforcement Learning". en. In:

arXiv:1803.00590 [cs, stat]. arXiv: 1803.00590. URL: http://arxiv.org/abs/1803.00590 (visited on 11/14/2019).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Lee, Sungjin, Qi Zhu, Ryuichi Takanobu, Zheng Zhang, Yaoqin Zhang, Xiang Li, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, et al. (2019). "ConvLab: Multi-Domain End-to-End Dialog System Platform". In: *ACL 2019*, p. 64.

Levin, Esther and Roberto Pieraccini (1997). "A stochastic model of computer-human interaction for learning dialogue strategies." In: *Eurospeech*. Vol. 97. Citeseer, pp. 1883–1886.

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* 8.1, pp. 11–23.

Levine, Sergey, Zoran Popovic, and Vladlen Koltun (2011). "Nonlinear inverse reinforcement learning with gaussian processes". In: *Advances in neural information processing systems* 24.

Li, Jinchao, Baolin Peng, Sungjin Lee, Jianfeng Gao, Ryuichi Takanobu, Qi Zhu, Minlie Huang, Hannes Schulz, Adam Atkinson, and Mahmoud Adada (2020). "Results of the multi-domain task-completion dialog challenge". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence, Eighth Dialog System Technology Challenge Workshop*.

Li, Xiujun, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen (2016). "A user simulator for task-completion dialogues". In: *arXiv preprint arXiv:1612.05688*.

Li, Yuxi (Oct. 2018). "Deep Reinforcement Learning". en. In: *arXiv:1810.06339 [cs, stat]*. arXiv: 1810.06339. URL: http://arxiv.org/abs/1810.06339 (visited on 10/09/2019).

Lipton, Zachary C., Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng (Nov. 2017). "BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems". en. In: *arXiv:1608.05081 [cs, stat]*. arXiv: 1608.05081. URL: http://arxiv.org/abs/1608.05081 (visited on 09/28/2020).

Litman, Diane J (1985). *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. Tech. rep. ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE.

Liu, Bing, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck (Apr. 2018). "Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems". In: *arXiv.org*. URL: http://arxiv.org/abs/1804.06512.

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". In: *arXiv preprint arXiv:1506.08909*.

Michini, Bernard and Jonathan P How (2012). "Improving the efficiency of Bayesian inverse reinforcement learning". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, pp. 3651–3656. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.310.4398&rep=rep1&type=pdf (visited on 11/14/2019).

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous Methods for Deep Reinforcement Learning". en. In: p. 10.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (Dec. 2013). "Playing Atari with Deep Reinforcement Learning". en. In: *arXiv:1312.5602 [cs]*. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602 (visited on 12/12/2019).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis (Feb. 2015). "Human-level control through deep reinforcement learning". en. In: *Nature* 518.7540, pp. 529–533. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14236. URL: http://www.nature.com/articles/nature14236 (visited on 12/12/2019).

Monti, Federico, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein (2019). "Fake news detection on social media using geometric deep learning". In: *arXiv preprint arXiv:1902.06673*.

Munos, Rémi, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare (Nov. 2016). "Safe and Efficient Off-Policy Reinforcement Learning". en. In: *arXiv:1606.02647 [cs, stat]*. arXiv: 1606.02647. URL: http://arxiv.org/abs/1606.02647 (visited on 08/03/2020).

Nachum, Ofir, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine (2019). "Why does hierarchy (sometimes) work so well in reinforcement learning?" In: *arXiv preprint arXiv:1909.10618*.

Ng, Andrew Y. and Stuart Russell (2000). "Algorithms for Inverse Reinforcement Learning". In: URL: http://ai.stanford.edu/~ang/papers/icml00-irl.pdf (visited on 11/14/2019).

Osband, Ian, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy (2016). "Deep Exploration via Bootstrapped DQN". en. In: p. 9.

Osband, Ian, Daniel Russo, and Benjamin Van Roy (2013). "(More) efficient reinforcement learning via posterior sampling". In: *Advances in Neural Information Processing Systems* 26.

Paek, Tim and Roberto Pieraccini (2008a). "Automating spoken dialogue management design using machine learning: An industry perspective". In: *Speech communication* 50.8-9, pp. 716–729.

— (Aug. 2008b). "Automating spoken dialogue management design using machine learning: An industry perspective". en. In: *Speech Communication* 50.8-9, pp. 716–729. ISSN: 01676393. DOI: 10.1016/j.specom.2008.03.010. URL: https:

//linkinghub.elsevier.com/retrieve/pii/S0167639308000344 (visited on 11/14/2019).

Papangelis, Alexandros and Yannis Stylianou (2019). "Single-model multi-domain dialogue management with deep learning". In: *Advanced Social Interaction with Agents*. Springer, pp. 71–77.

Parr, Ronald and Stuart Russell (1998). "Reinforcement learning with hierarchies of machines". In: *Advances in neural information processing systems*, pp. 1043–1049.

Pathak, Deepak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell (July 2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Honolulu, HI, USA: IEEE, pp. 488–489. ISBN: 978-1-5386-0733-6. DOI: 10.1109/CVPRW.2017.70. URL: http://ieeexplore.ieee.org/document/8014804/ (visited on 10/05/2020).

Peng, Baolin, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong (2017). "Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2231–2240.

Pietquin, Olivier (2013). "Inverse reinforcement learning for interactive systems". In: *Proceedings of the 2nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*. MLIS '13. event-place: Beijing, China. New York, NY, USA: ACM, pp. 71–75. ISBN: 978-1-4503-2019-1. DOI: 10.1145/2493525.2493529. URL: http://doi.acm.org/10.1145/2493525.2493529.

Pietquin, Olivier and Thierry Dutoit (2006). "A probabilistic framework for dialog simulation and optimal strategy learning". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.2, pp. 589–599.

Pietquin, Olivier and Helen Hastie (2013). "A survey on metrics for the evaluation of user simulations". In: *The knowledge engineering review* 28.1, pp. 59–73.

Piot, Bilal, Matthieu Geist, and Olivier Pietquin (2017). "Bridging the Gap Between Imitation Learning and Inverse Reinforcement Learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 28, pp. 1814–1826.

Rahimi, Afshin, Trevor Cohn, and Timothy Baldwin (2018). "Semi-supervised User Geolocation via Graph Convolutional Networks". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2009–2019.

Ramachandran, Deepak (2007). "Bayesian Inverse Reinforcement Learning". en. In: p. 6. (Visited on 11/14/2019).

Rastogi, Abhinav, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan (2020). "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 8689–8696.

Raux, Antoine, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi (2005). "Let's go public! taking a spoken dialog system to the real world". In: *in Proc. of Interspeech 2005*. Citeseer.

Rieser, Verena and Oliver Lemon (2011). *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

Ritter, Alan, Colin Cherry, and Bill Dolan (2010). "Unsupervised modeling of twitter conversations". In.

Rohmatillah, Mahdin and Jen-Tzung Chien (2021). "Corrective Guidance and Learning for Dialogue Management". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1548–1557.

Rojas-Barahona, Lina M. and Christophe Cerisara (2014). "Bayesian Inverse Reinforcement Learning for Modeling Conversational Agents in a Virtual Environment". en. In: *Computational Linguistics and Intelligent Text Processing*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Alexander Gelbukh. Vol. 8403. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 503–514. DOI: 10.1007/978-3-642-54906-9_41. URL: http://link.springer.com/10.1007/978-3-642-54906-9_41 (visited on 12/12/2019).

Ross, Stephane and J Andrew Bagnell (2010). "Efficient Reductions for Imitation Learning". en. In: p. 8. (Visited on 01/09/2020).

Ross, Stéphane and J.Andrew Bagnell (June 2014). "Reinforcement and Imitation Learning via Interactive No-Regret Learning". en. In: *arXiv:1406.5979 [cs, stat]*. arXiv: 1406.5979. URL: http://arxiv.org/abs/1406.5979 (visited on 01/09/2020).

Ross, Stéphane, Geoffrey Gordon, and J.Andrew Bagnell (2011). "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". en. In: *Proceedings of the Workshop on Artificial Intelligence and Statistics, AISTATS*, p. 9. (Visited on 12/13/2019).

Rummery, Gavin A and Mahesan Niranjan (1994). *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.

Sacks, Harvey, Emanuel A Schegloff, and Gail Jefferson (1978). "A simplest systematics for the organization of turn taking for conversation". In: *Studies in the organization of conversational interaction*. Elsevier, pp. 7–55.

Salimans, Tim and Richard Chen (Dec. 2018). "Learning Montezuma's Revenge from a Single Demonstration". en. In: *arXiv:1812.03381 [cs, stat]*. arXiv: 1812.03381. URL: http://arxiv.org/abs/1812.03381 (visited on 09/28/2020).

Sanchez-Gonzalez, Alvaro, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia (2020). "Learning to simulate complex physics with graph networks". In: *International Conference on Machine Learning*. PMLR, pp. 8459–8468.

Sanchez-Lengeling, Benjamin, Emily Reif, Adam Pearce, and Alexander B. Wiltschko (2021). "A Gentle Introduction to Graph Neural Networks". In: *Distill*. https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.

Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young (2007). "Agenda-based user simulation for bootstrapping a POMDP dialogue system". en. In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers on XX - NAACL '07*. Rochester, New York: Association for Computational Linguistics, pp. 149–152. DOI: 10.3115/1614108.1614146. URL: http://portal.acm.org/citation.cfm?doid=1614108.1614146 (visited on 01/13/2020).

Schatzmann, Jost, Karl Weilhammer, Matt Stuttle, and Steve Young (2006). "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies". In: *The knowledge engineering review* 21.2, pp. 97–126.

Schatzmann, Jost and Steve Young (2009). "The hidden agenda user simulation model". In: *IEEE transactions on audio, speech, and language processing* 17.4, pp. 733–747.

Schegloff, Emanuel A (1968). "Sequencing in conversational openings 1". In: *American anthropologist* 70.6, pp. 1075–1095.

Schrading, Nicolas, Cecilia Ovesdotter Alm, Raymond Ptucha, and Christopher Homan (2015). "An analysis of domestic abuse discourse on reddit". In: *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 2577–2583.

Schulman, John, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel (Apr. 2017a). "Trust Region Policy Optimization". en. In: *arXiv:1502.05477 [cs]*. arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477 (visited on 12/20/2019).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (Aug. 2017b). "Proximal Policy Optimization Algorithms". en. In: *arXiv:1707.06347 [cs]*. arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347 (visited on 01/06/2020).

Searle, John R and John Rogers Searle (1969). *Speech acts: An essay in the philosophy of language*. Vol. 626. Cambridge university press.

Shah, Pararth, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck (2018). "Building a conversational agent overnight with dialogue self-play". In: *arXiv preprint arXiv:1801.04871*.

Stalnaker, Robert C (1978). "Assertion". In: *Pragmatics*. Brill, pp. 315–332.

Stokes, Jonathan M, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae,

Zohar Bloom-Ackermann, et al. (2020). “A deep learning approach to antibiotic discovery”. In: *Cell* 180.4, pp. 688–702.

Strens, Malcolm (2000). “A Bayesian framework for reinforcement learning”. In: *ICML*. Vol. 2000, pp. 943–950.

Su, Pei-Hao, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young (July 2017). “Sample-efficient Actor-Critic Reinforcement Learning with Supervised Data for Dialogue Management”. en. In: *arXiv:1707.00130 [cs]*. arXiv: 1707.00130. URL: http://arxiv.org/abs/1707.00130 (visited on 12/18/2019).

Su, Pei-Hao, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young (June 2016a). “Continuously Learning Neural Dialogue Management”. en. In: *arXiv:1606.02689 [cs]*. arXiv: 1606.02689. URL: http://arxiv.org/abs/1606.02689 (visited on 12/19/2019).

— (June 2016b). “On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems”. en. In: *arXiv:1605.07669 [cs]*. arXiv: 1605.07669. URL: http://arxiv.org/abs/1605.07669 (visited on 11/05/2020).

Sutton, Richard (1988). “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3.1, pp. 9–44.

Sutton, Richard and Andrew Barto (2018). *Reinforcement learning: an introduction*. en. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-03924-6.

Sutton, Richard, David McAllester, Satinder Singh, and Yishay Mansour (2000). “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. en. In: *Advances in Neural Information Processing Systems (NIPS) 12*, pp. 1057–1063. (Visited on 12/19/2019).

Sutton, Richard, Doina Precup, and Satinder Singh (Aug. 1999). “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. en. In: *Artificial Intelligence* 112.1-2, pp. 181–211. ISSN: 00043702. DOI: 10.1016/S0004-3702(99)00052-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370299000521 (visited on 11/22/2019).

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* 112.1-2, pp. 181–211.

Takanobu, Ryuichi, Hanlin Zhu, and Minlie Huang (Aug. 2019a). “Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog”. en. In: *arXiv:1908.10719 [cs]*. arXiv: 1908.10719. URL: http://arxiv.org/abs/1908.10719 (visited on 07/16/2020).

— (2019b). “Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 100–110.

Takanobu, Ryuichi, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang (2020). “Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation”. en. In: p. 14.

Tang, Da, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li, and Tony Jebara (2018). "Subgoal Discovery for Hierarchical Dialogue Policy Learning". en. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2298–2309. DOI: 10.18653/v1/D18-1253. URL: http://aclweb.org/anthology/D18-1253 (visited on 11/20/2019).

Tetreault, Joel and Diane Litman (2006). "Using reinforcement learning to build a better model of dialogue state". In: *11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 289–296.

Traum, David R (1999). "Speech acts for dialogue agents". In: *Foundations of rational agency*. Springer, pp. 169–201.

Ultes, Stefan, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, and Milica Gasic (2017a). "Pydial: A multi-domain statistical dialogue system toolkit". In: *Proceedings of ACL 2017, System Demonstrations*, pp. 73–78.

Ultes, Stefan, Lina M. Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, and Steve Young (2017b). "PyDial: A Multi-domain Statistical Dialogue System Toolkit". en. In: *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, pp. 73–78. DOI: 10.18653/v1/P17-4013. URL: http://aclweb.org/anthology/P17-4013 (visited on 11/14/2019).

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903*.

Walker, Marilyn and Steve Whittaker (1990). "Mixed initiative in dialogue: an investigation into discourse segmentation". In: *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pp. 70–78.

Wang, Huimin, Baolin Peng, and Kam-Fai Wong (2020). "Learning Efficient Dialogue Policy from Demonstrations through Shaping". en. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6355–6365.

Wang, Yaqing, Quanming Yao, James T Kwok, and Lionel M Ni (2020). "Generalizing from a few examples: A survey on few-shot learning". In: *ACM computing surveys (csur)* 53.3, pp. 1–34.

Wang, Zhuoran, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou (2015). "Learning domain-independent dialogue policies via ontology parameterisation". In: *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 412–416.

Wang, Ziyu, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas (July 2017). "Sample Efficient Actor-Critic with Experience Replay". en. In: *arXiv:1611.01224 [cs]*. arXiv: 1611.01224. URL: http://arxiv.org/abs/1611.01224 (visited on 12/13/2019).

Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas (2016). "Dueling Network Architectures for Deep Reinforcement Learning". en. In: *International conference on machine learning*, pp. 1995–2003.

Watkins, Christopher John Cornish Hellaby (1989). "Learning from delayed rewards". In.

Weisz, Gellért, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić (Feb. 2018). "Sample Efficient Deep Reinforcement Learning for Dialogue Systems with Large Action Spaces". en. In: *arXiv:1802.03753 [cs, stat]*. arXiv: 1802.03753. URL: http://arxiv.org/abs/1802.03753 (visited on 12/18/2019).

Wen, Tsung-Hsien, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young (2016). "A network-based end-to-end trainable task-oriented dialogue system". In: *arXiv preprint arXiv:1604.04562*.

Wen, Zheng, Doina Precup, Morteza Ibrahimi, Andre Barreto, Benjamin Van Roy, and Satinder Singh (2020). "On efficiency in hierarchical reinforcement learning". In: *Advances in Neural Information Processing Systems* 33, pp. 6708–6718.

Wesselmann, Paula, Yen-Chen Wu, and Milica Gasic (May 2019). "Curiosity-driven Reinforcement Learning for Dialogue Management". en. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, pp. 7210–7214. ISBN: 978-1-4799-8131-1. DOI: 10.1109/ICASSP.2019.8683033. URL: https://ieeexplore.ieee.org/document/8683033/ (visited on 10/05/2020).

Williams, Jason (2007). "Partially observable Markov decision processes for spoken dialogue management". PhD thesis. University of Cambridge.

Williams, Jason, Antoine Raux, and Matthew Henderson (2016). "The dialog state tracking challenge series: A review". In: *Dialogue & Discourse* 7.3, pp. 4–33.

Williams, Jason, Antoine Raux, Deepak Ramachandran, and Alan Black (2013). "The dialog state tracking challenge". In: *in Proc. of the SIGDIAL 2013 Conference*, pp. 404–413.

Williams, Jason and Steve Young (2005). "Scaling up POMDPs for Dialog Management: The"Summary POMDP"Method". In: *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.* IEEE, pp. 177–182.

— (2007). "Partially observable Markov decision processes for spoken dialog systems". In: *Computer Speech & Language* 21.2, pp. 393–422.

Williams, Ronald (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3, pp. 229–256.

Wittgenstein, L (1953). "Philosophical Investigations. Oxford: Basil Blackwell". In.

Wu, Zonghan, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip (2020). "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1, pp. 4–24.

Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner (Mar. 2016). "Maximum Entropy Deep Inverse Reinforcement Learning". en. In: *arXiv:1507.04888 [cs]*.

arXiv: 1507.04888. URL: http://arxiv.org/abs/1507.04888 (visited on 11/14/2019).

Xu, Keyulu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka (2018). "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826*.

Ye, Fanghua, Jarana Manotumruksa, and Emine Yilmaz (2021). "Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation". In: *arXiv preprint arXiv:2104.00773*.

Young, Steve, Milica Gasic, Blaise Thomson, and Jason Williams (May 2013a). "POMDP-Based Statistical Spoken Dialog Systems: A Review". en. In: *Proceedings of the IEEE* 101.5, pp. 1160–1179. ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2012.2225812. URL: http://ieeexplore.ieee.org/document/6407655/ (visited on 12/11/2019).

Young, Steve, Milica Gašić, Blaise Thomson, and Jason Williams (2013b). "Pomdp-based statistical spoken dialog systems: A review". In: *Proceedings of the IEEE* 101.5, pp. 1160–1179.

Yu, Lantao, Tianhe Yu, Chelsea Finn, and Stefano Ermon (2019). "Meta-Inverse Reinforcement Learning with Probabilistic Context Variables". en. In: p. 12.

Zang, Xiaoxue, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen (July 2020). "MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines". In: *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Association for Computational Linguistics, pp. 109–117.

Zhang, Shangtong, Hengshuai Yao, and Shimon Whiteson (2021). "Breaking the deadly triad with a target network". In: *International Conference on Machine Learning*. PMLR, pp. 12621–12631.

Zhang, Zheng, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu (2020). "Recent advances and challenges in task-oriented dialog systems". In: *Science China Technological Sciences*, pp. 1–17.

Zhifei, Shao and Er Meng Joo (2012). "A review of inverse reinforcement learning theory and recent advances". In: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, pp. 1–8. URL: https://ieeexplore.ieee.org/abstract/document/6256507/ (visited on 11/14/2019).

Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun (2020). "Graph neural networks: A review of methods and applications". In: *AI Open* 1, pp. 57–81.

Zhu, Qi, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang (2020). "ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 142–149.

Ziebart, Brian D, Andrew Maas, J Andrew Bagnell, and Anind K Dey (2008). "Maximum Entropy Inverse Reinforcement Learning". en. In: p. 6. (Visited on 11/14/2019).

Personal publications

Cordier, Thibault, Tanguy Urvoy, Lina Maria Rojas-Barahona, and Fabrice Lefèvre (2020). "Diluted Near-Optimal Expert Demonstrations for Guiding Dialogue Stochastic Policy Optimisation". In: *Proceedings of the Workshop of Human in the Loop Dialogue Systems, NeurIPS 2020*. URL: https://sites.google.com/view/hlds-2020/home.

— (2022a). "Et la robustesse ? ... bordel ! Comment les stratégies de dialogue par apprentissage structuré résistent aux bruits des entrées ?" In: *Proceedings of Journées d'Études sur la Parole – JEP 2022*, pp. 501–510. URL: https://www.isca-speech.org/archive/jep_2022/cordier22_jep.html.

— (2022b). "Graph Neural Network Policies and Imitation Learning for Multi-Domain Task-Oriented Dialogues". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2022, Edinburgh, UK, 07-09 September 2022*. Ed. by Oliver Lemon, Dilek Hakkani-Tür, Junyi Jessy Li, Arash Ashrafzadeh, Daniel Hernández García, Malihe Alikhani, David Vandyke, and Ondrej Dusek. Association for Computational Linguistics, pp. 91–100. URL: https://aclanthology.org/2022.sigdial-1.10.

— (2023). "Few-Shot Structured Policy Learning for Multi-Domain and Multi-Task Dialogues". In: *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*. Ed. by Andreas Vlachos and Isabelle Augenstein. Association for Computational Linguistics, pp. 432–441. URL: https://aclanthology.org/2023.findings-eacl.32.

AVIGNON UNIVERSITÉ

Hierarchical Imitation and Reinforcement Learning for Multi-Domain Task-Oriented Dialogue Systems

par
Thibault CORDIER

Résumé : Dans cette thèse de doctorat, nous étudions les systèmes de dialogue orientés tâches qui sont des systèmes conçus pour aider les utilisateurs à accomplir des tâches spécifiques, telles que la réservation d'un vol ou d'un restaurant. Ils s'appuient généralement sur un paradigme d'apprentissage par renforcement pour modéliser le dialogue permettant au système de raisonner sur les objectifs et les préférences de l'utilisateur, et de sélectionner les actions qui conduiront au résultat souhaité.

Malgré les avancées récentes, les systèmes de dialogue orientés tâches présentent encore plusieurs limites. L'une d'entre elles est la tendance de ces systèmes à échouer lorsque les utilisateurs s'écartent du comportement attendu ou introduisent de nouveaux objectifs au milieu de la conversation. Un autre problème est la difficulté de concevoir des systèmes robustes capables de gérer un large éventail de tâches.

Nous nous concentrons spécifiquement sur l'apprentissage à partir d'un nombre limité d'interactions, ce qui est crucial en raison de la rareté et du coût des interactions humaines. Les algorithmes standards d'apprentissage par renforcement nécessitent généralement une grande quantité de données d'interaction pour obtenir de bonnes performances. Pour relever ce défi, nous visons à rendre les systèmes de dialogue plus efficaces en termes d'échantillonnage dans leur entraînement.

Nous nous sommes inspirés principalement des idées d'imitation et de hiérarchie. Notre première contribution explore l'intégration de l'imitation dans l'apprentissage par renforcement. Nous nous appuyons sur la littérature existante qui souligne l'importance de l'imitation dans l'apprentissage, car les humains apprennent souvent en imitant des experts qui possèdent des connaissances précieuses. Nous étudions comment utiliser efficacement les démonstrations d'experts pour extrapoler les connaissances avec un effort de généralisation minimal. Alors que l'imi-

tation s'avère efficace pour obtenir des performances et tirer parti de démonstrations réussies, nous observons des limites lorsqu'il s'agit de traiter une complexité plus élevée, en particulier dans le cadre d'un dialogue orienté tâches multi-domaines.

Notre deuxième contribution porte sur l'exploitation de la hiérarchie et de la structure inhérentes aux dialogues. En nous inspirant de l'avantage que présente la décomposition de problèmes complexes en problèmes plus simples, nous explorons la manière d'exploiter les similitudes entre les tâches et les domaines dans les systèmes de dialogue. En décomposant le problème principal en tâches élémentaires que nous maîtrisons, nous tirons parti de la hiérarchie pour résoudre efficacement des problèmes plus vastes et plus complexes. Cette approche permet d'économiser du temps de formation en partageant des stratégies entre des tâches similaires.

Enfin, nous consolidons nos résultats précédents et soulignons l'importance de l'apprentissage à partir d'un petit nombre d'interactions humaines dans les applications du monde réel. Les techniques d'apprentissage efficaces sur le plan de l'échantillonnage sont essentielles dans ce contexte, et nos recherches portent sur le développement de solutions efficaces dans le cadre de nos découvertes précédentes.

Mots-clés : système de dialogue orienté tâche, apprentissage par renforcement, apprentissage par imitation, apprentissage hiérarchique, stratégie structurée

Abstract : In this Ph.D thesis, we study task-oriented dialogue systems that are systems designed to assist users in completing specific tasks, such as booking a flight or ordering food. They typically rely on reinforcement learning paradigm to model the dialogue that allows the system to reason about the user's goals and preferences, and to select actions that will lead to the desired outcome.

Despite these advances, there are still several limitations to task-oriented dialogue systems. One issue is the tendency of such systems to fail when users deviate from expected behavior or introduce new goals mid-conversation. Another issue is the difficulty of designing robust systems that can handle a wide range of tasks.

Our focus is specifically on learning from a limited number of interactions that is crucial due to the scarcity and costliness of human interactions. Standard reinforcement learning algorithms typically require a large amount of interaction data to achieve good performance. To address this challenge, we aim to make dialogue systems more sample-efficient in their training.

To guide our contribution journey, we draw from two main ideas: imitation and hierarchy. Our first contribution explores the integration of imitation with reinforcement learning. We build upon existing literature that emphasises the importance of imitation in learning, as humans often learn by imitating experts who possess valuable knowledge. We investigate how to effectively use expert demonstrations to extrapolate knowledge with minimal generalisation effort. While imitation proves efficient for achieving performance and leveraging successful trajectories, we observe limitations when dealing with higher complexity, particularly in multi-domain task-oriented dialogue.

Our second contribution focuses on harnessing the hierarchy and structure inherent in dialogues. Taking inspiration from the advantage of decomposing complex problems into simpler

ones, we explore how to exploit task and domain similarities in dialogue systems. By decomposing the main problem into elementary tasks that we master, we leverage hierarchy to solve larger and more complex problems efficiently. This approach saves training time by sharing policies across similar tasks.

Lastly, we consolidate our previous findings and emphasise the importance of learning from a small number of human interactions in real-world applications. Sample-efficient learning techniques are essential in this context, and our investigation revolves around developing effective solutions within the framework of our previous discoveries.

Keywords : task-oriented dialogue system, reinforcement learning, imitation learning, hierarchical learning, structured policy

LABORATOIRE
INFORMATIQUE D'AVIGNON