Personalised Dialogue Management for Users with Speech Disorders

Iñigo Casanueva

Department of Computer Science
University of Sheffield
United Kingdom

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 50 figures.

Iñigo Casanueva
Wednesday 22nd February, 2017

To my parents and Nadia

Acknowledgements

First I would like to thank my two supervisors, Phil and Thomas. I am really grateful to Phil for giving me the opportunity to do a PhD in this great group and supporting me from the beginning. I am really grateful to Thomas for always finding time in his really busy schedule for helping me and giving me invaluable guidance in my research. I am also very grateful to Heidi, who during my first years was my "third supervisor". I also want to thank most of the people from SpandH and the MINI group, especially Ricard, Jose and Erfan for the very helpful discussions we had and Morrie, Ray and Oscar for fixing the grid whenever I broke it.

In the personal side, first I would like to thank my parents, which have always been supporting me even if I did not have time to give them all the attention they deserve. Second, I would like to thank Nadia for all the support she has given me in the worst moments of this four years, this thesis would not have been possible without her. Third, I would like to thank Jenny and Noe for standing me during these four years even if I was not the best of the housemates. I also want to thank my brother Carlos for giving me these invaluable five advices, and Mireia, Haritz, Aiora and the new one for making me have such a good time every time I visited them. I would also like to thank Son Gohan for teaching me to never give up. During these four years I have made so many friends that made my time in Sheffield much better and I am really grateful to all of them, but it would be impossible to list them all here, so I will just mention the two I have probably spent most time with (and the ones that have heard me complain the most): Asier and Dario. Finally, I also want to thank my friends in Irun, that even if I do not go back very often, they always make me feel as if I had never left.

Abstract

Many electronic devices are beginning to include Voice User Interfaces (VUIs) as an alternative to conventional interfaces. VUIs are especially useful for users with restricted upper limb mobility, because they cannot use keyboards and mice. These users, however, often suffer from speech disorders (e.g. dysarthria), making Automatic Speech Recognition (ASR) challenging, thus degrading the performance of the VUI. Partially Observable Markov Decision Process (POMDP) based Dialogue Management (DM) has been shown to improve the interaction performance in challenging ASR environments, but most of the research in this area has focused on Spoken Dialogue Systems (SDSs) developed to provide information, where the users interact with the system only a few times. In contrast, most VUIs are likely to be used by a single speaker over a long period of time, but very little research has been carried out on adaptation of DM models to specific speakers.

This thesis explores methods to adapt DM models (in particular dialogue state tracking models and policy models) to a specific user during a longitudinal interaction. The main differences between personalised VUIs and typical SDSs are identified and studied. Then, state-of-the-art DM models are modified to be used in scenarios which are unique to long-term personalised VUIs, such as personalised models initialised with data from different speakers or scenarios where the dialogue environment (e.g. the ASR) changes over time. In addition, several speaker and environment related features are shown to be useful to improve the interaction performance. This study is done in the context of homeService, a VUI developed to help users with dysarthria to control their home devices. The study shows that personalisation of the POMDP-DM framework can greatly improve the performance of these interfaces.

Glossary

$\mathcal{S}$ Set of dialogue states.

$\mathcal{A}$ Set of machine actions.

Ω Set of observations.

s A dialogue state from the set $\mathcal{S}$.

a A machine action from the set $\mathcal{A}$.

ω An observation from the set Ω.

u A user action.

$\mathbf{b}$ Belief state, distribution over the set of dialogue states $\mathcal{S}$.

R Reward function, $R(\mathbf{b}, a) \to \mathcal{R}$.

r Immediate reward given by the reward function, $r_i = R(\mathbf{b}_i, a_i)$.

γ Discount factor, $0 \leq \gamma \leq 1$.

c Accumulated reward, sum of immediate rewards from timestep i to the end of the dialogue, discounted by γ, $c_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$.

Q Q-function, expected c_i when starting from belief state $\mathbf{b}_i$, taking action a_i and then following policy π, $Q^{\pi}(\mathbf{b}_i, a_i) = E[c_i]$. π is usually omitted for clarity.

π Dialogue policy, $\pi(\mathbf{b}) \to \mathcal{A}$.

$\mathbf{X}$ Set of observed belief-action points.

$\mathbf{Z}$ Set of observed temporal difference points.

$\mathbf{U}$ Set of inducing temporal difference points.

$\mathbf{x}$ Belief-action point, $(\mathbf{b}, a)$.

$\mathbf{z}$ Temporal difference point, two consecutive belief-action points $(\mathbf{b}_i, a_i, \mathbf{b}_{i+1}, a_{i+1}, \gamma_i)$.

k Kernel function in the belief action space, computes covariance between two belief-action points, $k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

k^a Action kernel function, computes covariance between two action points, $k^a_{i,j} = k^a(a_i, a_j)$.

k^b Belief kernel function, computes covariance between two belief points, $k^b_{i,j} = k^b(\mathbf{b}_i, \mathbf{b}_j)$.

k^{td} Temporal difference kernel function, computes covariance between two temporal difference points, $k^{td}_{i,j} = k^{td}(\mathbf{z}_i, \mathbf{z}_j)$.

$\mathbf{K}$ Gram matrix, $\mathbf{K}^{kx}_{I,J}$ matrix of covariances between the set of points I and J computed with the kernel kx.

$\mathbf{H}$ Band-diagonal matrix defining the temporal difference relationship between consecutive belief-action points.

$\bar{Q}$ Mean of the Q-function when modelled as a Gaussian process.

$\hat{Q}$ Variance of the Q-function when modelled as a Gaussian process.

Q^{mc} Q-function modelled with a Monte-Carlo approach.

Q^{td} Q-function modelled with a temporal difference approach.

Q^{dtc} Q-function modelled with a temporal difference approach and the deterministic training conditional approximation.

$\mathbf{s}$ Vector of environment features.

k^s Environment kernel function, computes covariance between two environment vectors, $k^s_{i,j} = k^b(\mathbf{s}_i, \mathbf{s}_j)$.

$\mathbf{o}$ Dialogue features. Usually composed by an N-best list output plus the last system action.

$\mathbf{o}_s$ Environment features used as input to the dialogue state tracker.

$\mathbf{o}^t_v$ Value specific dialogue features.

$\mathbf{s}^t_v$ Value specific environment features.

Acronyms

ADA	Adaptation data amount.
APW	Accuracy per word.
ASR	Automatic Speech Recognition.
DM	Dialogue Management.
DST	Dialogue State Tracking.
DSTC	Dialogue State Tracking Challenge.
DTC	Deterministic Training Conditional.
GP	Gaussian Process.
hS	homeService.
IV	I-Vector.
MC	Monte-Carlo.
MDP	Markov Decision Process.
PO	Policy Optimization.
POMDP	Partially Observable Markov Decision Process.
RL	Reinforcement Learning.
RNN	Recurrent Neural Network.
SDS	Spoken Dialogue System.
SLU	Spoken Language Understanding.

SU Simulated User.

TD Temporal Difference.

TL Transfer Learning.

VUI Voice User Interface.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the emergence of "smart" personal devices during the last decade (smart-phones, smart-TVs, car-control, etc.) the need for functional voice interfaces to control these devices has become a priority in speech technology research. Predominantly, these devices will have a single user (or very few) who will interact many times over a long period of time, in some sense building a "relationship" with the user. Due to the remarkable advances in Automatic Speech Recognition (ASR) and speech synthesis, many "personal assistant" applications (Siri, Google Now, Cortana, Alexa, etc.) have been fielded, but most users still rely on the more conventional keyboard based interfaces (Pieraccini et al., 2009). This limited take-up of speech interfaces might be due to the poor performance of the device's Spoken Language Understanding (SLU) module, and to the Dialogue Management (DM), the module in charge of controlling the interaction (i.e. deciding what action to take after a user command, etc.). These modules (especially the DM), still rely mostly on rule-based methods which make them inflexible, costly to maintain and difficult to adapt to novel situations. Statistical data-driven SLU and DM have shown promising behaviour in constrained domains but there are scalability problems in implementing them in multi-domain systems (Tur and De Mori, 2011, Young et al., 2013).

Due to the influence of commercial interests, research in speech technologies is mainly focused on commercial applications like the ones mentioned above. These interfaces can make the control of smart devices easier for their users, but will not affect a big change in the user's lives. However, Voice User Interfaces (VUIs) can make a life-changing difference to people with disabilities caused by conditions such as motor neuron disease, cerebral palsy or a stroke. Such people are unable to use conventional keyboard-screen-mouse interfaces. Functional voice-enabled environmental control interfaces would allow this group of people to perform everyday actions such as controlling home devices (e.g. TV, lights) or browsing the internet in an easy way. The problem of using state-of-the-art speech technologies in this area is that physical disabilities and speech disorders often co-occur, because they have a common underlying cause. For example, people with motor neuron diseases will often have a speech impairment known as dysarthria, which will make conventional ASR unusable for them, and subsequently all the modules of the voice interface which depend on the ASR.

Users with speech disorders have very special speech characteristics which vary with the severity of the condition. This makes user adaptation a key aspect when developing voice interfaces for this group of users. Speaker adaptation has been shown to greatly improve ASR performance both for conventional and dysarthic users (Christensen et al., 2012a), but very little effort has been devoted to adapt DM models to a specific user. This is mainly because most DM-related research has been studied in the context of information gathering Spoken Dialogue Systems (SDSs) (Raux et al., 2005, Young et al., 2010), which users call to ask for information about a specific topic (e.g. hotels in a city). In this kind of system, many different users interact with the same SDS and most of them will do it once or a few times, so adaptation is not feasible. In contrast to this, both personal assistants and VUIs for people with disabilities can greatly benefit from adapting their dialogue models to their individual users.

1.1 Motivations

Reinforcement Learning (RL), in the form of the Partially Observable Markov Decision Process (POMDP) framework, has been shown to be a very promising approach to spoken DM (Young et al., 2013), joining the power for long term planning of Markov Decision Processes (MDPs) with an increased robustness against ASR errors. However, very little work has been done up to this date on the adaptation of POMDP dialogue models to specific users, even though it is a topic with many potential applications. The main motivation of this thesis is to explore the possibility of adapting the dialogue manager of a VUI that interacts with a single user over a long period of time, in the context of interfaces designed to assist disabled people with dysarthria. Adaptation for the models in charge of the two main tasks of a dialogue manager, Dialogue State Tracking (DST) and Policy Optimization (PO), is studied. In addition, the thesis also aims to identify and address several issues that arise from applying POMDP-based DM to long-term personal interaction with assistive VUIs.

1.2 Research outline

When the POMDP framework is applied to information gathering SDSs, several assumptions are made. For example, it is assumed that all the dialogue interaction data will come from one distribution, even if the data is collected from several speakers. It is also assumed that the environment dynamics will be stable and will not change over time. If the POMDP framework is applied to assistive VUIs for dysarthric speakers, where a single speaker will interact with the system over a long period of time, these assumptions do not hold. For example, data coming from two different dysarthric speakers will follow different distributions. Moreover, if the ASR is adapted online with user specific data collected through interaction, the environment dynamics will change. Therefore, existing POMDP models need to be modified to be applied in assistive VUIs.

This thesis researches the development of methods based on existing POMDP models applicable to assistive VUIs for dysarthric speakers. It also develops methods to personalise the models to the speakers and to adapt them to the changes in the environment. The research is done in the context of the homeService (hS) project, a project carried out at the university of Sheffield, which aims to take state-of-the-art speech technologies to the users homes, with most of the users suffering from severe disabilities and dysarthria. The main questions this thesis aims to answer are the following:

- Can the POMDP dialogue management framework improve the interaction performance of a VUI developed for dysarthric speakers? If so, what modifications are necessary?

- Can the adaptation of DM models to specific speakers or environments improve the dialogue interaction performance?

- Can the reinforcement learning DM framework be applied in an environment in which the ASR performance changes over a long period of time?

- Can POMDP models developed for a single dysarthric speaker be trained with data coming from different dysarthric speakers?

- Can acoustic and environment related information be used to improve the performance of personalised VUIs? Can this information improve the generalisation to unseen situations?

1.3 Contributions

The first and maybe the most important contribution of this thesis is that this is the first work to address the issue of adaptation of POMDP-based DM systems to a specific user who interacts with the system over a long period of time. The trends in speech technology suggest that personal spoken interfaces (in the form of command based VUIs or natural language based personal assistants) will be a very important research topic in future years. This thesis analyses the improvement that can be obtained by using POMDP-based DM in these interfaces, addresses the issues that might arise because of the differences with typical SDSs used in DM research and proposes modifications to existing models to make them usable in long-term personalised environments. In addition, this thesis makes several more specific contributions to the DM field:

- **Analysis of adaptive ASR for dysarthric users and dysarthric user simulation**: Interacting with users (especially disabled ones) to get data for training or evaluation is extremely expensive (Christensen et al., 2015). Simulation of the user and the ASR environment can reduce the necessary effort to develop and evaluate the systems (Schatzmann et al., 2006). However, the ASR behaviour with dysarthric speakers can be difficult to simulate, especially if the acoustic models are adapted with different amounts of speaker specific data. In this thesis, a simulation environment for hS systems is developed, composed of a simulation model for the user behaviour and a simulation model for the ASR. To simulate the ASR, the recognition accuracy of different dysarthric speakers when different amounts of adaptation data are available are analysed and different speaker-ASR simulation environments are created.

- **Integration of POMDPs in command based VUIs**: The POMDP DM framework has been successfully applied to information retrieval SDSs, but it is not clear if it can benefit simpler systems such as command based VUIs. However, the robustness against high ASR errors provided by the POMDP framework, as well as its adaptability, could be of great help when interacting with dysarthric speakers. In this thesis a tractable POMDP framework for command based environmental control interfaces is developed, showing that it increases the interaction performance in comparison to the typically used rule-based dialogue managers.

- **Development of the temporal difference kernel framework**: State-of-the-art, model-free, reinforcement learning algorithms such as Gaussian Process

(GP)-based RL need to use approximation methods to maintain the system tractability. However, the approximation methods proposed so far do not allow arbitrary data selection, a necessary requirement for some hS scenarios. To solve this, the GP-RL equations are modified by defining a kernel function in the *temporal difference* space. The resulting equations not only permit a wide range of approximation methods that allow data selection to be applied, but also simplify the previous equations, making it easier to understand the model.

- **Policy optimization in varying environments**: In systems such as hS, the ASR performance will vary as more acoustic data is collected and the acoustic models are adapted. This will lead to a mismatch between training and evaluation conditions that has to be dealt with. A method to improve the policy optimization in this scenario is developed, based on the definition of similarities between different environments.

- **Transfer learning between different speakers**: Another scenario that is very likely to be found is the setting up of a system for a new user while only data from other speakers is available for training. It is shown that initialising a system with data from different speakers, while defining a similarity metric between speakers, greatly boosts the dialogue policy performance in a newly set-up system. In addition, the proposed method permits online adaptation of the policy model as dialogue data becomes available through interaction with the user.

- **Feature augmented dialogue state tracking**: Historically, the features used as input for DM had to be as low dimensional and uncorrelated as possible to maintain the tractability of the generative model based DST. Newly proposed discriminative models for DST open up the possibility to use higher dimensional, possibly correlated, input features. Features extracted from the acoustics and the ASR are proposed, showing to improve DST generalisation to unseen speakers and unseen dialogue states.

Although these contributions are evaluated in the context of assistive VUIs for dysarthric speakers, they are potentially applicable to any kind of personalised dialogue management application, such as personal assistants or in-car control systems.

1.4 Thesis overview

Chapter 2 gives an overview of the state-of-the-art techniques used to model the components required to build an assistive VUI for dysarthric speakers. Firstly, existing approaches to voice-enabled assistive control interfaces are briefly introduced. Secondly, automatic speech recognition and understanding is introduced with a focus on acoustic model adaptation to disordered speech. Then, the state of the art of the main topic of the thesis is introduced: data-driven statistical dialogue management, with special emphasis on POMDP-based dialogue management, dialogue state tracking and policy optimization algorithms.

Chapter 3 introduces the environment in which the proposed techniques will be evaluated, the *homeService environment*, and identifies its differences to typical SDS environments. The chapter continues by proposing a framework for evaluating POMDP-based dialogue management, developing a simulated hS environment and proposing a tractable POMDP architecture for this environment. The chapter ends by evaluating POMDP-based DM in this environment and identifying some necessary modifications needed to operate in long-term, single user, varying environment VUIs.

Chapter 4 begins by analysing one of the main problems of applying state-of-the-art model-free POMDP models to long-term personalised VUIs: the impossibility of arbitrary selection and discarding of data in tractable methods. Then, two methods that deal with this problem are proposed and analysed, showing that the *temporal difference kernel* is the most promising of them. This method is then further evaluated in two scenarios in which data selection is necessary: varying ASR environments and transfer learning between speakers. A method to weight the data coming from different environments by defining environment features is also proposed.

In chapter 5, the environment features which showed to be a promising approach in the previous chapter are tested as input features for discriminative DSTs. It is shown how discriminative trackers can get more benefit from these features, improving the dialogue state tracking generalisation to unseen speakers. A SDS architecture less restricted than the typical pipeline architecture is also proposed. A second set of experiments explores how features giving information about the ASR behaviour, extracted from the phonetic structure of the commands, can improve the generalisation to unseen dialogue states.

Chapter 6 gives the main conclusions and lists the achievements of the thesis. In the last part of the chapter, the fast changes that have occurred in research in dialogue management during the last decade are discussed, and an insight of the future directions it might take (or is already taking) is given.

1.5 Publications

Most of the work presented in this thesis has been published in international peer-reviewed conferences. In total, four papers were published as main author and two more had contributions as second author. In addition, two of the publications as main author were nominated for best paper awards. The list of publications obtained while this doctoral work was done is:

- H. Christensen, **I. Casanueva**, S. Cunningham, P. Green, and T. Hain, *Home-Service: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition*, in Proceedings of the 4th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT), 2013. (Christensen et al., 2013b)

- **I. Casanueva**, H. Christensen, T. Hain, and P. Green, *Adaptive speech recognition and dialogue management for users with speech disorders*, in Proceedings of INTERSPEECH, 2014. (Nominated for best student paper award) (Casanueva et al., 2014)

- H. Christensen, **I. Casanueva**, S. Cunningham, P. Green, and T. Hain, *Automatic selection of speakers for improved acoustic modelling: recognition of disordered speech with sparse data*, in 2014 IEEE Spoken Language Technology Workshop (SLT), 2014. (Christensen et al., 2014)

- **I. Casanueva**, T. Hain, H. Christensen, Ricard Marxer and P. Green, *Knowledge transfer between speakers for personalised dialogue management*, in Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2015. (Nominated for best paper award) (Casanueva et al., 2015)

- **I. Casanueva**, T. Hain and P. Green, *Improving generalisation to new speakers in spoken dialogue state tracking*, in Proceedings of INTERSPEECH, 2016. (Casanueva et al., 2016a)

- **I. Casanueva**, T. Hain, M. Nicolao and P. Green, *Using phone features to improve dialogue state tracking generalisation to unseen states*, in Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2016. (Casanueva et al., 2016b)

A high percentage of the work described in the following chapters is based on the work presented in these papers. The sections of chapters 2 and 3 describing the hS project are partially based on the work published in Christensen et al. (2013b) and Christensen et al. (2014). Most of the sections in chapter 3 are based on the work published in Casanueva et al. (2014). Several sections presented in chapter 4 are based on the work published in Casanueva et al. (2015). The models presented in chapter 5 have been published in Casanueva et al. (2016a) and Casanueva et al. (2016b).

Chapter 2

Data driven methods for assistive spoken interfaces

Due to the increased ageing of the population (the aged population is currently at its highest level in human history, and is expected to continue increasing (UN report, 2002)), the demand for technologies to improve the life quality amongst the elderly and the physically impaired is increasing. Electronic assistive technology can improve the ability to live independently, especially for users with restricted upper limb mobility. For this group of users, speech can be a very attractive input to control these assistive interfaces (Clark and Roemer, 1977, Cohen and Graupe, 1980, Hawley, 2002, Hawley et al., 2007a,b). The interest in spoken interfaces to interact with electronic devices (often called Voice User Interfaces (VUIs) (Pieraccini and Huerta, 2005) or Spoken Dialogue Systems (SDSs) (Jokinen and McTear, 2009)), has increased during the last decade, leading to improvements in the performance of these interfaces. Applying the advances obtained in this field to spoken electronic assistive technology could greatly improve its performance.

Spoken interfaces in the form of VUIs or SDSs interact with the user using voice commands or natural language in a turn taking fashion. They are composed of several modules, shown in figure 2.1, which can be clustered into 3 groups:

- The first group, sometimes called Spoken Understanding System (SUS), has two main components: The Automatic Speech Recognition (ASR) and the Spoken Language Understanding (SLU) module. The objective of the SUS is to process the input (in the form of speech) and convert it into a high level representation that the machine can "understand". It could be seen as a front end or feature extractor, which converts a high dimensional input (speech) into a lower dimensional one (concepts)

- The second module, known as dialogue manager, is in charge of the inter-

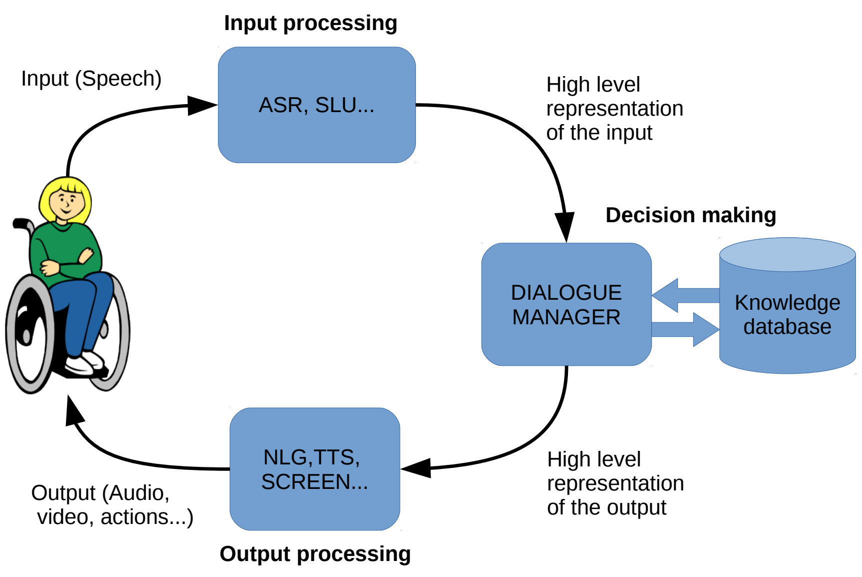

Figure 2.1: *Diagram showing the main components of an assistive VUI. The arrows represent the data flow and the rectangles the modules that process the data.*

action decisions. It maintains a representation of the dialogue history (all the information seen so far in the dialogue) called *dialogue state*. In each dialogue turn, this module takes as input the output of the SUS, updates its dialogue state, and given the new dialogue state decides the next action to take. This action might require the extraction of information from a database (e.g. checking the hotels in a certain area). The action taken is outputted in a high level representation similar to the input of the dialogue manager.

- The third module converts the high level representation of the Dialogue Management (DM) action into the output signal. This signal is often in the form of synthetic speech (an answer), but it can also be a physical action such as showing a document on the screen, turning the lights on, or any action that a machine can perform.

VUIs and SDSs are often seen as different concepts in the literature. SDSs are seen as task oriented dialogue systems in where all the interaction is based on speech and the system can output some piece of information relevant for the user (e.g. bus schedule information (Raux et al., 2005) or tourist information (Young et al., 2010)) in the form of synthetic speech. VUIs, on the other hand, are seen as speech interfaces for substituting the typical keyboard based interfaces of the electronic devices (Cohen et al., 2004). A VUI can be seen as an instance of a SDS, where the natural language used for the interaction is easier (even just single commands) and the answers of the system often come as actions performed by the controlled devices. In any case, the research done in SDSs can be applied to VUIs and vice-versa.

However, when applying voice controlled assistive technologies to users with disabilities, an important problem often arises: disabilities leading to restricted upper-limb mobility often co-occur with speech disorders such as dysarthria (Duffy, 2013), degrading the performance of the ASR (Borrie et al., 2012, Mengistu and Rudzicz, 2011). Personalisation of the ASR to the user has been proven to improve the accuracy (Christensen et al., 2012a, Sharma and Hasegawa-Johnson, 2010, 2013). However, to further improve the performance of the interfaces for these users, the personalisation of the different modules of the VUI could also be necessary, but this has not been widely studied so far.

In this chapter, the state of the art of the modules necessary to build a data-driven assistive VUI is reviewed. The first section discusses the current approaches in voice controlled assistive interfaces. This is followed by a brief review of speech recognition and understanding along with its adaptation to disordered speech in sections 2.2 and 2.3. Section 2.4 focuses on reviewing the main topic of this thesis: dialogue management. Then, the two main components of data-driven dialogue management, dialogue state tracking and policy optimisation, are reviewed in sections 2.5 and 2.6 respectively. Finally, section 2.7 reviews the main dialogue evaluation and user simulation methods.

2.1 Speech technologies for assisted living

Typical interfaces for electronic devices (such as remote controls, keyboards or mouse devices) can be difficult to use for people with physical disabilities. In addition, each device uses a different interface, which might be counter intuitive and require time to learn. Environmental control interfaces allow the use of a single, easy to use, interface to control several aspects of the home environment (e.g. TV, radio or lights). Typically, these systems are operated using a switch-scanning interface which accommodates the limited motor control abilities of the users, but for severely disabled users, these interfaces involve a lot of effort and are very time consuming. Speech-enabled interfaces can provide an attractive alternative way of accessing digital devices, as they would reduce the effort and time needed (Clark and Roemer, 1977, Cohen and Graupe, 1980, Hawley, 2002). The success of such interfaces is highly dependent on the recognition accuracy that can be achieved at the time of deployment; if the performance is too poor, the user is likely to lose interest and will not be motivated to use the system.

However, due to their physical disability, a significant proportion of people requiring electronic assistive technology suffer from the speech disorder known as dysarthria (Duffy, 2013). Dysarthric speakers suffer a loss of control of the speech articulators, affecting the quality of their speech production. As a result of this, inexperienced listeners can find speech from dysarthric speakers difficult to understand (Borrie et al., 2012, Mengistu and Rudzicz, 2011). In addition, ASR systems are also affected, making automatic speech recognition infeasible for the severe cases of dysarthria (Rosen and Yampolsky, 2000, Rudzicz, 2011).

During the last decade, several projects have tried to improve the ASR accuracy with dysarthric speakers to be used as input for environmental control interfaces (Christensen et al., 2012b, Hawley et al., 2007a,b). However, most of these systems were developed in relatively small scale studies with the main focus being on the observed ASR performance. Porting such systems and set-ups to more "realistic" scenarios presents a greater challenge, especially because of the larger number of users involved and the need for a large degree of automation, whilst still accommodating the needs of the individual users for personalisation. Recently, the homeService (hS) project (Christensen et al., 2013b) developed speech-driven electronic control interfaces and evaluated them in more realistic scenarios: the users' homes.

2.1.1 The homeService project

homeService (hS) is a project developed in the University of Sheffield[1], where dysarthric users with restricted upper limb mobility are provided with a speech-driven environmental control interface or VUI to give spoken access to other digital applications. Several dysarthric users have been recruited for a longitudinal study (Christensen et al., 2015) in which each user is engaged with homeService for at least 6 months.

In hS, researchers work with the users in a collaborative way; the users effectively become part of the research team. This had been done in previous similar projects (Christensen et al., 2012b, Hawley et al., 2007a) which included user requirement studies (Schuler and Namioka, 1993). In this process, the users inform the design and specifications for their personal hS system, such as the devices they want to control or which commands they want to use to control the devices. In

[1]homeService was part of the UK EPSRC Project Natural Speech Technology, a collaboration between the Universities of Edinburgh, Cambridge and Sheffield. `http://www.natural-speech-technology.org/`.

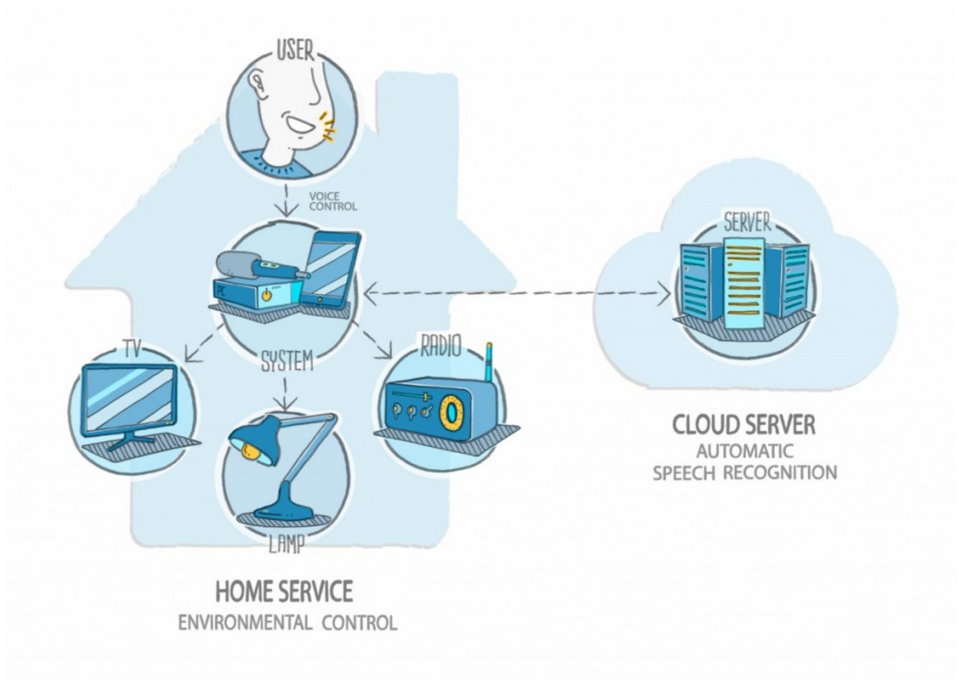

Figure 2.2: *Artistic representation of a homeService system.*

addition, the researchers work with the users to close what it has been referred to as the "virtuous circle" (Christensen et al., 2013b). An initial "operating point" is established for each user, a task which is sufficiently simple so that good performance from the ASR can be expected and yet sufficiently useful that the user's interest is maintained. Practice through interaction with the system improves the user's pronunciation consistency and, more importantly, provides more data for ASR training or adaptation. The data collected can be then used to improve the ASR acoustic models. When the performance of the system has improved sufficiently, the vocabulary and range of devices homeService controls can be widened.

Another improvement of hS with respect to previous assistive spoken interfaces, is that the ASR runs remotely "in-the-cloud" and is connected to the homeService users' home by a dedicated broadband link (see figure 2.2). This approach enables the researchers to collect speech data, change vocabularies, experiment with adaptation algorithms, train new statistical models, etc. without having to modify the equipment in the user's home. This enables rapid system modification and reduces the amount of researcher time spent travelling to the users' homes. This permits the new models to be deployed when they are ready and the new data to be analysed as soon as it is collected.

A central idea of this project is the notion that the user, through interacting with the system, will provide additional speech data. This data can be used to improve the acoustic models, which should in turn help motivate the user to use the system more, collecting more speech data – closing the *virtuous circle*. Two different data collection strategies are employed in hS. Firstly, the initial data collection phase or *enrolment phase*, collects data through basic recordings. When sufficient enrolment data has been collected, this data is used to adapt a set of speaker independent models to the speech of the user before the system is deployed. From then on the data collection takes place as the user interacts with the system. At regular intervals this data is used to update the models so the system continues to tune into the particular characteristics of the user's voice as well as their environment.

However, only the ASR module of the hS VUI is personalised to the user. Other modules such as the DM follow simple rule-based approaches to control the interaction. One of the main objectives of this thesis is to introduce state-of-the-art dialogue managers in hS-like systems and adapt the dialogue models to the user. This is done in chapters 3, 4 and 5.

2.2 Automatic Speech Recognition

The objective of ASR is to convert an acoustic input (speech) into a sequence of words (text) (Gales and Young, 2008). Statistical ASR tries to find the probability of a given sequence of words $\mathbf{w}$ given a sequence of acoustic features $\mathbf{X}$, $P(\mathbf{w}|\mathbf{X})$. The sequence of acoustic features $\mathbf{X}$ is extracted from the audio input in a pre-processing step (Davis and Mermelstein, 1980, Hermansky, 1990). Using Bayes' rule, $P(\mathbf{w}|\mathbf{X})$ can be decomposed as:

$$P(\mathbf{w}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{X})} \tag{2.1}$$

Where $P(\mathbf{X}|\mathbf{w})$ is the acoustic likelihood of the acoustic features given the word sequence and $P(\mathbf{w})$ is the prior probability of the sequence of words. $P(\mathbf{X})$ is the prior probability of the acoustic features, which is independent of the sequence of words, becoming a normalisation constant. However, as the number of possible word sequences that can be found can be very large, often ASRs only do *decoding*, finding the most likely sequence of words $\hat{\mathbf{w}}$ as:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \left(P(\mathbf{X}|\mathbf{w})P(\mathbf{w})\right) \tag{2.2}$$

Equation 2.2 if often considered as the "fundamental equation" of ASR. It shows the two main components of an ASR system: the acoustic model $P(\mathbf{X}|\mathbf{w})$ and the language model $P(\mathbf{w})$. Both acoustic modelling and language modelling are important parts of modern statistically-based speech recognition algorithms.

A language model computes the prior probability of a word sequence. This is usually approximated by conditioning the probability of each word in the previous $N-1$ words. These are called *N-Gram* language models (Jurafsky, 2000). Recently, recurrent neural network based language models (Mikolov et al., 2010) have improved the performance by increasing the context window on which the probability of the word is conditioned.

An acoustic model computes the likelihood of an acoustic feature sequence given the sequence of words. Since speech has temporal structure and can be encoded as a sequence of spectral vectors spanning the audio frequency range, hidden Markov models (HMMs) provide a natural framework for constructing such models. Usually, Gaussian mixture models (GMMs) are used to estimate the HMM state probabilities. Recently the field has benefited from advances in deep learning by using deep neural networks to extract more sophisticated feature vectors (Hermansky and Fousek, 2005) and to directly replace GMMs to model the observation probabilities of each state (Bourlard and Morgan, 2012, Deng et al., 2013, Hinton et al., 2012). This has greatly increased the performance of ASR.

As the number of words in a vocabulary of any language can be very high, to reduce the search space of a speech recogniser, acoustic subunits called "phones" are used. There are around 40 phones in English. The corresponding acoustic model is synthesised by concatenating phone models to make words as defined by a pronunciation dictionary (Richmond et al., 2010).

As both acoustic models and language models are trained from data, the performance of the speech recognition engine will depend greatly on the difficulty of the task (vocabulary size) and on the amount of in-domain training data. Speech recognition in under resourced domains such as highly-technical domains (Fox et al., 2013), under resourced languages (Harper, 2014) or disordered speech (Deller et al., 1991, Rosen and Yampolsky, 2000) can be challenging.

For some tasks (including DM), it is important not only to estimate the most likely sequence of words, but also the N-best list of most likely hypotheses with their corresponding confidence scores (Jiang, 2005). The confidence scores give an indication of the likelihood that the recogniser attaches to each word sequence. Ideally, these confidence scores should be the posterior probabilities of each word sequence, but, as the number possible word sequences can be huge, pruning is used during decoding to maintain the search to the most likely sequences only. Therefore, the system only outputs the likelihood of each word sequence. Many modern ASR systems can output a *word lattice* (Murveit et al., 1994), a directed graph that efficiently encodes possible word sequences. Word-lattices may be converted into N-best lists by normalising the likelihoods by the probability mass given in all the lattice (Evermann and Woodland, 2000a,b).

2.2.1 Disordered speech recognition

The term dysarthria refers to a range of speech disorders arising from the loss of control of the speech articulators. It affects 170 per 100,000 of the population, being the most common speech disorder (Duffy, 2013). The underlying causes can be acquired neurological conditions as a result of stroke or traumatic brain injury or congenital conditions such as cerebral palsy. People with severe dysarthria can be close to unintelligible to unfamiliar listeners, though they can generally communicate successfully with family and friends (Mengistu and Rudzicz, 2011). There are established assessment procedures for speech and language therapists to determine the dysarthria severity (Enderby, 1980).

In cases of low to moderate dysarthria severity, state-of-the-art, speaker independent ASR systems can have a reasonable performance. For people with severe dysarthria, however, these systems have been shown to be less successful (Hawley, 2002, Rosen and Yampolsky, 2000). The first attempts to apply ASR to dysarthric speech were small scale studies done using conventional ASR techniques (Deller et al., 1991), not showing very promising results. One of the main issues faced by these systems was the lack of dysarthric data. Recently, the release of some larger corpora has enabled the research on applying more sophisticated ASR techniques to dysarthric speech, such as speaker adaptation.

Dysarthric speech corpora

The increasing performance of speech recognition technology has been directly related to the use of larger corpora, but until recently only the Nemours database (Menendez-Pidal et al., 1996) was commonly available for research into dysarthric ASR. This corpus contains a total of 814 sentences from 11 male speakers with mild to moderate dysarthria severity. In the last decade, however, the available dysarthric data has increased with the appearance of the UASpeech (Kim et al., 2008) and TORGO (Rudzicz et al., 2012) databases. These corpora are still orders of magnitude below the corpora used in modern ASR; UASpeech has around 18 hours and the TORGO recordings amount to 23 hours, not all of which is disordered speech. In contrast, it is normal to train ASR systems in hundreds of hours of normal speech (Carletta et al., 2005), whereas ASRs used in industry can be trained in thousands of hours (Graves and Jaitly, 2014). Nevertheless UAspeech and TORGO opened the possibility to apply some modern training and adaptation techniques to dysarthric speech.

The limited amount of training data makes researchers and clinical scientists face several problems when deploying ASR systems for dysarthric users. People with physical disabilities which also affect their speech often find it difficult to contribute a sufficient amount of data; for some dysarthric speakers supplying a few minutes of data can be very tiring and also lead to distress, especially for people with degenerative illnesses such as Parkinson's disease. Tools to help with the collection of data and the deployment of assistive technologies can greatly help speech professionals. In this context, the *CloudCAST initiative* (Green et al., 2015, 2016) was recently launched, trying to make remote adaptive technology available to professionals who work with speech; such as therapists, educators and clinicians.

Speaker adaptation

As it has been previously mentioned, standard speaker independent ASR systems do not have an acceptable performance with severe dysarthric speakers. Speaker dependent speech recognition has been shown to be more appropriate for these users. This is because speaker dependent models are trained directly with the speaker's utterances rather than assuming their speech is similar to typical speech. Speaker dependent recognisers have shown a promising performance with severely dysarthric users in several studies (Hawley et al., 2007a,b), but with very small input vocabularies, which can limit the potential usefulness of the system.

The new corpora of dysarthric speech released in recent years (UASpeech and TORGO), have enabled researchers to conduct more systematic studies (Christensen et al., 2012a, 2013a, Sharma and Hasegawa-Johnson, 2010, 2013), comparing different techniques using reference test sets. In these studies, speaker adaptation of acoustic models was shown to be a more promising approach than directly training the models with speaker specific data. Adaptation techniques, such as Maximum Likelihood Linear Regression (Gales, 1998) and Maximum A Posteriori (MAP) (Gauvain and Lee, 1992), have been used in ASR to tune speaker independent models to the speech of an individual. Using this adaptation techniques speaker dependent systems can be built, but by using only a relatively small amount of adaptation data. In Christensen et al. (2012a), MAP adaptation was shown to be a successful way of establishing acoustic models for dysarthric speakers. This work presented ASR accuracy results on the UASpeech task using about 40 minutes of data for each speaker employing a vocabulary of 455 words.

2.3 Spoken Language Understanding

After the acoustic input has been processed by the ASR, the next step is to understand the meaning of the recognised string of words. The SLU module (sometimes called semantic decoder) (Tur and De Mori, 2011, Wang et al., 2005) performs this task, extracting the semantics of an utterance given the output of the ASR. To constrain the semantics the system can understand, a set of *concepts* is usually defined. This set is defined differently in each domain, making SLU a domain specific task. Many SDSs define the set of concepts using a shallow level of semantics called *dialogue acts* (Stolcke et al., 2000, Young, 2007), designed to capture just enough meaning in an utterance to facilitate a dialogue. In summary, an SLU module takes a sentence as input and gives a concept (or dialogue act) as output.

Historically, SLU modules used hand-crafted semantic template grammars to extract semantic concepts and discern the dialogue act (Aust et al., 1995, Ward and Issar, 1994). This is an effective technique for small scale dialogue systems, where the semantic space is simple enough to hand-craft the rules to understand it. This is the case for assistive VUIs, where the rules are often as simple as a mapping from commands to concepts (Casanueva et al., 2014). In larger scale dialogue systems, however, multiple iterations of user-testing are needed before achieving adequate coverage. Data driven SLU approaches can be a better option in this cases.

If the possible set of concepts can be enumerated in a short list, the SLU can be treated as a multi-class classification task, where a single multi-class classifier is trained. However, if the concept output has some structure, these classifiers would need to enumerate all the possible structures, causing sparsity in training examples. Modern data driven SLU approaches (Henderson et al., 2012, Mairesse et al., 2009, Mesnil et al., 2013, Vukotic et al., 2015) attempt to exploit the structure to simplify the classification, in what it is often called *slot-filling* tasks. These systems factorise the set of possible output concepts to a set of *slots*, each of which can take a value from a set of mutually exclusive values. Then, the system tries to assign a value to each slot given the ASR output observed. A good example of slot filling SLU is the ATIS evaluation task (Price, 1990). ATIS is the most popular resource to investigate data-driven SLU (it has been used for more than 25 years), providing a corpus in the flight reservation domain. However, this corpus is very small compared with today's machine learning standards and only includes single turn utterances. The French *Media* corpus (Bonneau-Maynard et al., 2005) provides a larger corpus in a more challenging hotel reservation domain with multi-turn interactions. Being in French, however, can be a problem for some research groups and has reduced its popularity. Industry research has been able to carry studies in more challenging SLU domains, but by using corpora that is not publicly available (Hakkani-Tür et al., 2016, Liu et al., 2015).

2.4 Data driven spoken Dialogue Management

Dialogue is a decision making process. In each turn, a dialogue system has to analyse the user's utterance, and given this utterance and the information it has already collected through the dialogue, decide which action to take. The ASR and SLU "analyse" the user utterance converting it to a semantic representation, so the next step is to use this representation (and all the previous ones) to decide the action that should be taken next. The Dialogue Manager (DM)[1] is the module in charge of this (Young et al., 2013). In other words, the dialogue manager is the component responsible for the control and flow of the dialogue. In each dialogue turn, it decides the appropriate output given the dialogue history (all the observations seen during the interaction so far). The dialogue manager could be seen as the *brain* of an SDS: given the inputs it receives from its sensors (the ASR and SLU), analyses what has happened in the dialogue so far, and decides which is the best action to take next.

In the previous paragraph, the concept of dialogue history appears a couple of times. The dialogue history refers to all the information that has been observed by the DM during all the dialogue. This includes all the information passed to the DM by the input modules[2] of the system up to the current turn, as well as the decisions taken by the DM so far. The dialogue history is usually encoded in a fixed dimensional stationary representation called the *dialogue state*, which tries to collect all the relevant information happened in the dialogue in a compact way (Young, 2000). However, to cope with the uncertainty coming from possible ASR errors as well as the inherent uncertainty in natural language, many systems work with a probability distribution over the dialogue state called the *belief state* (Roy et al., 2000). The dialogue manager then uses this representation to decide which action to take the next turn. The function that maps the dialogue state or belief state to an action is called the *dialogue policy*.

Therefore, dialogue management can be divided into two main tasks: Dialogue State Tracking (DST) and Policy Optimization (PO). DST updates the dialogue

[1]Depending on the context, the acronym DM can refer to *Dialogue Management* or to *Dialogue Manager*

[2]The DM usually takes as input the output of the SLU, but some systems can get as input features extracted from other modules such as the ASR (Henderson et al., 2014c,d) or other feature extractors (Casanueva et al., 2016a,b).

state or belief state in each turn. PO models the decision process to decide which is the optimal[1] action to take given the current dialogue state.

Most of currently deployed computer based dialogue managers use rule-based policies following different approaches (Goddeau et al., 1996, Larsson and Traum, 2000, Lucas, 2000, McTear, 1998, Sutton et al., 1996). The main advantage of these systems is that they can be built without collecting training data; the knowledge of an expert is enough. These policies are a good approach for small scale SDSs with good ASR performance, because the number of possible situations (or dialogue states) to take into account is small, so the set of rules to follow in each situation can be hand-coded. This approach, however, has several shortcomings: it does not scale to more complex systems, it has difficulties dealing with noisy ASR systems, it is not easily adaptable or transferable to other domains, tuning them to new environments is costly, etc. In addition, these systems require a lot of effort to fine tune them: after the systems have been built, they need to be tested with users and then manually adjusted to improve user satisfaction. This process is repeated iteratively until a suitable level of user satisfaction is reached (Pieraccini et al., 2009).

Machine learning methods are a more suitable approach for scalable DM. As DST can be modelled as a sequential classification task, supervised learning techniques have been shown to be a good approach (Henderson, 2015a, Thomson, 2013). For PO, however, supervised learning has several flaws. In principle, supervised learning techniques offer a way of learning the dialogue policy directly from data when given a large enough dialogue corpus, but, due to the large number of distinct dialogue states that can occur, even a very large dialogue corpus would represent only a tiny fraction of the total set of plausible dialogues. Even if the system's policy can be learnt in a supervised fashion, it would be restricted to imitating a particular behaviour at a particular dialogue state. It is not guaranteed that such behaviour would be the optimal or lead to a successful dialogue (Levin et al., 2000).

Policy optimization is not a classification problem, it is a decision making problem, or an *optimal control of stochastic dynamic systems* problem. The machine learning response to this problem is the Reinforcement Learning (RL) paradigm (Sutton and Barto, 1998). RL was proposed for dialogue management almost 20 years ago (Levin et al., 1998, 2000, Singh et al., 1999, Young, 2000, 2002), and the research on this approach has continued since then, getting increasingly promising results (Gašić and Young, 2014, Henderson et al., 2014a, Raux et al., 2005, Williams and

[1]Optimality in some dialogue domains can be hard to define. In task oriented dialogues, however, it can be defined as a trade-off between dialogue success rate and time spent. Section 2.7.1 reviews this in more detail.

Young, 2007, Williams et al., 2013, Young et al., 2010). With an RL approach, the dialogue is modelled as a sequential decision process and the policy is optimised with respect to an objective measure of dialogue performance. A reward is assigned to each turn which depends on the dialogue state and action taken by the system, and RL tries to maximise the sum of rewards for all the dialogue. In contrast to supervised learning approaches, a dialogue manager using reinforcement learning can explore new (and possibly better) behaviours. It is therefore able to choose a strategy which optimises the overall performance as defined by the objective measure[1].

2.4.1 Dialogue management as a Partially Observable Markov Decision Process

In RL (Sutton and Barto, 1998), an *agent* learns from interaction with an *environment* (which might be unknown), getting feedback from the environment in the form of *immediate rewards*. The agent-environment interaction cycle is presented in figure 2.3. In each discrete time-step t, the agent has a perception of the environment, represented by a state s_t. Based on this state, the agent takes an action a_t, which affects the environment changing its state to s_{t+1} while the agent observes an immediate reward r_{t+1}. Then the cycle is repeated. The aim is to take the optimal sequence of actions to maximise the expected long-term *accumulated reward* or *return*, the sum of rewards for each time-step.

As it was mentioned in the previous section, task oriented spoken dialogue can be modelled as an RL problem. On one side, the agent is defined as the dialogue manager. On the other side, the environment is defined as the user plus the information passing channels: the ASR and the SLU for the input processing, and the response generation system for the output processing (see figure 2.1). The state of the environment is defined as the dialogue state and the actions are defined as the set of machine actions that the agent (dialogue manager) can take (e.g. questions, database queries, showing documents in the screen, etc.). Each discrete time step t is set as each dialogue turn, and the immediate rewards are defined in order to maximise the dialogue success rate whilst minimizing the dialogue length (the number of turns needed to complete the dialogue).

The state of the agent (the dialogue state) depends on all previously visited states, as well as the actions it has taken. To reduce the complexity, the state can be assumed to depend only on the previous state and action, satisfying the

[1]It is worth to mention though that, recently, supervised learning approaches to policy learning have been shown to be useful to bootstrap an initial policy, which is later optimised using RL (Fatemi et al., 2016, Su et al., 2016a).

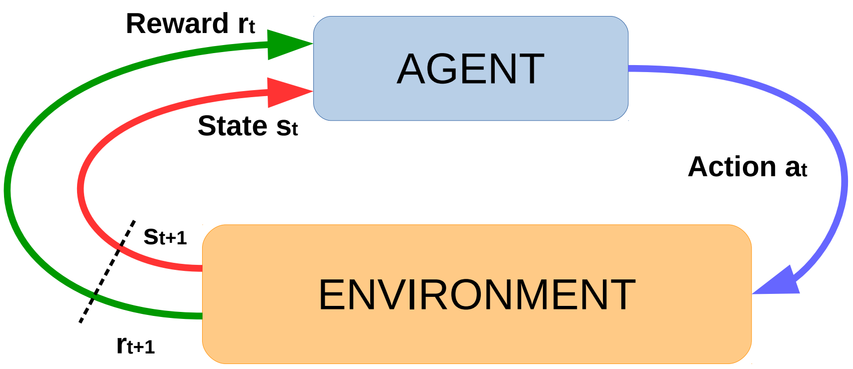

Figure 2.3: *Diagram showing the reinforcement learning loop.*

Markov property. If, in addition, the state is considered to be fully observable (the agent knows in every moment the true state of the environment), the system can be modelled as a Markov Decision Process (MDP) (Puterman, 2014). MDPs provide a mathematical framework for modelling decision making in situations where outcomes are partly random (environment) and partly under the control of a decision maker (actions). A MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ where:

- $\mathcal{S}$ is the set of states,
- $\mathcal{A}$ is the set of actions,
- T is the transition function,
- R is the reward function,
- γ is the discount factor.

At each time step, the environment is in some state $s \in \mathcal{S}$. The agent takes an action $a \in \mathcal{A}$, which causes the environment to transition to state s' with probability $T(s, a, s') = P(s' = s_{t+1} | s = s_t, a = a_t)$. Finally, the agent receives a reward equal to $R(s, a) \in \mathcal{R}$. Then the process repeats.

The core problem of MDPs is to find a *policy* π for the decision maker: a function that specifies the action $\pi(s) = a$ that the decision maker will choose

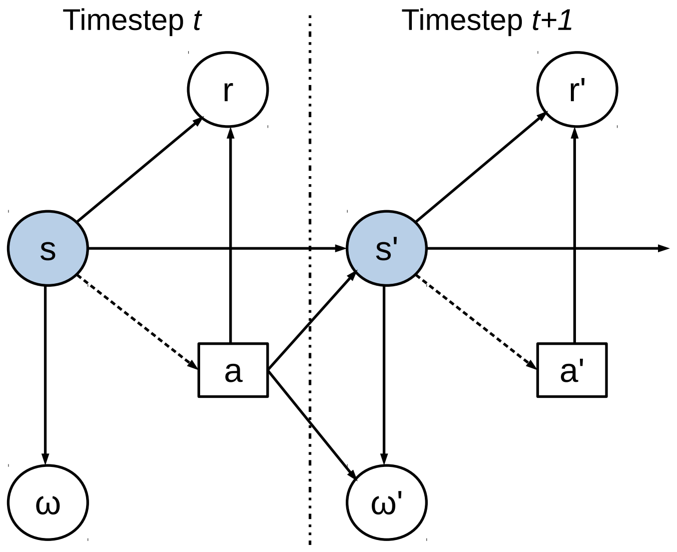

Figure 2.4: *POMDP influence diagram. Circles represent random variables and squares decisions taken by the policy. Shaded nodes represent unobserved variables and non shaded circles observed ones. solid lines indicate direct influence and dashed lines that a distribution of the variable (the belief state) is used.*

when in state s. The policy is optimised to maximise some cumulative function of the random rewards, typically the expected discounted sum over a potentially infinite horizon:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{2.3}$$

A discounting factor $1 \geq \gamma \geq 0$ is used to favour policies that generate high rewards sooner rather than later in the decision process. A discounting factor equal to 1 can only be used in tasks that are certain to finish in a finite number of steps, called episodic tasks. Dialogue management is one example of an episodic task.

The MDP framework, however, assumes that the set of states is fully observable, but this assumption does not hold in dialogue. In the perception that the dialogue manager has from the environment there is uncertainty coming from errors in the ASR and the understanding system. There are also inherent ambiguities in the human natural language (e.g. sarcasm). Therefore, the output of the SLU is a noisy observation of the true user intentions, so the system cannot be certain about the true dialogue state. The Partially Observable Markov Decision Process (POMDP) framework (Kaelbling et al., 1998) tackles this issue by assuming that the dialogue state is a latent variable that can be indeed inferred from noisy *observations* seen from the environment (Roy et al., 2000). Therefore, the dialogue manager works with the *belief state*, a probability distribution over the set of dialogue states (Rapaport, 1986).

Figure 2.4 shows the influence diagram of a POMDP. A POMDP is an extension of an MDPs defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$ where:

- $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ is an MDP
- Ω is the set of observations,
- O is the observation function,

In each turn, instead of seeing the true dialogue state, the dialogue manager receives an observation $\omega' \in \Omega$ which depends on the new state of the environment with probability given by $O(\omega', s', a) = P(\omega' = \omega_{t+1} | a = a_t, s' = s_{t+1})$. The belief state $\mathbf{b}$ needs to be inferred from this observation and the previous belief state. As the state s is not observable, the dialogue policy becomes a function of the belief state[1] $\pi(\mathbf{b}) = a$. Therefore, POMDP-based DM can be decomposed into two steps: The aforementioned step of estimating the optimal policy, named PO, and the estimation of the belief state, usually called belief update or DST in the dialogue management community. In the next section, the most successful DST approaches are described. In section 2.6, dialogue PO algorithms are surveyed.

2.5 Dialogue State Tracking

The dialogue state is a high level representation of all the information observed so far during the dialogue by the dialogue manager. This information usually includes user intention information (SLU output) and the actions taken by the DM so far, but it can be extended to include environment information, speaker information, ASR information or any information that might be relevant to decide which is the best action to take next. Due to the uncertainty coming from the information channels (ASR, SLU...), the dialogue state s is not directly observable. Instead, in each turn the dialogue manager sees an *observation* $\omega \in \Omega$, and uses this observation and the previous ones (as well as any other information seen during the dialogue) to estimate the *belief state* $\mathbf{b}$, a probability distribution over the set of states $\mathcal{S}$. The estimation of the belief state is known as DST. DST incorporates system outputs, user speech, context from previous turns, and other external information. If the belief state is defined as $\mathbf{b} = (b(s_1), b(s_2), ..., b(s_{|S|}))$, where

[1]Technically, a POMDP can be seen as a continuous state MDP in where the continuous state space lies in a simplex over the original set of states.

$b(s_i)$ represents the probability of the environment being in state s_i, then in each turn t:

$$b(s) = P(s = s_t|\mathbf{h}_t) \tag{2.4}$$

where $\mathbf{h}_t$ is the dialogue history until turn t, all the relevant information observed so far during the dialogue (e.g. $\mathbf{h}_t = (\omega_0, \omega_1, ...\omega_t, a_0, a_1, ..., a_{t-1})$ where each ω is an N-best output of the SLU and each a is an action taken by the dialogue manager. The dialogue history can include more information though).

Therefore, the task of DST is to estimate the distribution over the dialogue states $\mathbf{b}$ in each turn. There has been an increase of interest in DST following the Dialogue State Tracking Challenges (DSTCs) (Henderson et al., 2014a,b, Kim et al., 2016, Williams et al., 2013), with a substantial number of papers published on the subject since 2013.

2.5.1 Dialogue state set design

In the previous sections the dialogue states were defined as a set, $\mathcal{S}$, in which each state s represents a unique dialogue situation, encoding all the relevant information seen so far. However, the number of possible different situations that can occur during a dialogue is immense (typically exponential with respect to the size of the system (Gašić, 2011)). Therefore, the design of dialogue systems requires the (typically hand-crafted) effort of designing a tractable state space. This design should trade off between creating a set of states that can differentiate between the most important dialogue situations, whilst maintaining the size of this set as small as possible. To do so, task oriented SDSs are typically framed in terms of slots (sometimes called slot-based dialogue systems) (Henderson et al., 2014a, Williams et al., 2013). The set of slots and possible slot values are derived from an *ontology* which defines the systems domain, i.e. the scope of what it can talk about and the tasks that it can help the user complete. The ontology informs the set of possible actions the system can take, the possible semantics of the user utterances, etc. The set of dialogue states is then derived from combining the possible values for the different slots.

2.5.2 Dialogue State Tracking approaches

Early spoken dialogue systems used hand-crafted rules for DST, keeping a single top hypothesis for each slot of the dialogue state (Larsson and Traum, 2000, Zue et al., 2000). Such systems required no data to implement, and provided an accessible method for designers to incorporate expert knowledge of the dialogue domain. However, these systems were unable to make use of the entire SLU N-best lists, not accounting for uncertainty in a principled way. More recently, rule-based approaches able to track multiple hypotheses have also been proposed, using a small set of hand-crafted rules to compute the belief state given observations from the SLU (Kadlec et al., 2014, Wang and Lemon, 2013).

The first approaches to use machine learning methods for DST used generative approaches to model the dialogue state as a dynamic Bayesian network where the true state s is treated as an unobserved random variable (Thomson, 2013, Williams et al., 2005). Bayesian inference is then used to give an updated distribution over s given the system act a and a noisy observation of the user intent ω. Early generative approaches used exact inference, enumerating all possible dialogue states (Horvitz and Paek, 1999, Roy et al., 2000). The belief update equation of typical POMDP models (Kaelbling et al., 1998) is an example of generative DST:

$$b(s') = k \cdot O(\omega', s', a) \sum_{s \in \mathcal{S}} T(s, a, s') b(s) \tag{2.5}$$

where $O(\omega', s', a)$ and $T(s, a, s'))$ are the *observation model* and *transition model* of the POMDP respectively. k is a normalisation constant. This approach, however, is quadratic with the number of states, so it is usually intractable as the number of states can be very large. As a result, two approximations are typically used; either maintaining a beam of candidate dialogue states (Williams, 2007b, Young et al., 2010), or assuming conditional independence between components of the dialogue state (Thomson and Young, 2010, Williams and Young, 2007, Williams et al., 2005). Generative models, however, have several deficiencies (Williams, 2012). Generative approaches need to model all the correlations in the input features, so they cannot easily exploit arbitrary but potentially useful features observed in the dialogue history. These features would have to be included into the dynamic Bayesian network, requiring the learning or specification of new structures and dependencies. Many independence assumptions are made by generative models used for DST in order to make the models tractable. For example, many implementations do not model ASR and SLU error correlations, instead assuming independence for simplicity (Henderson, 2015b).

Contrary to generative modes that need to model the joint probability of the input features and the dialogue states, discriminative models directly model the conditional probability over dialogue states, given the input features. Therefore, there is no need to maintain the input feature space as small and uncorrelated as possible to maintain tractability, letting discriminative models to use larger and possibly correlated input features. The first attempt to build a discriminative dialogue state tracker was presented in Bohus and Rudnicky (2006), but it wasn't until the DSTCs were held (Henderson et al., 2014a, Williams et al., 2013) that the real potential of discriminative state trackers was shown. The DSTCs provided a common testbed to compare different DST models. The first two DSTCs were won by discriminative models which used a very large set of engineered features which captured the dialogue history (Lee, 2013, Williams, 2014). The third DSTC was won by a Recurrent Neural Network (RNN) based discriminative state tracker which directly took as input the ASR output. The first three DSTCs mainly showed the potential of discriminative models over the previously used generative models. Discriminative methods for DST can be split into two categories: static classifiers that encode the dialogue history in the input features, and sequence models that explicitly model dialogue as a sequential process.

In static state trackers, the probability over dialogue states in a given turn is conditioned on a feature representation of the whole dialogue history up to that point. Following the DSTCs, Maximum Entropy and neural network classifiers have been the most popular static models (Henderson et al., 2013, Lee, 2013, Metallinou et al., 2013, Ren et al., 2014, Sun et al., 2014, Williams, 2013). In order to classify arbitrary length sequences of dialogue states, which may change from turn to turn, static models must design special feature functions to encode the information seen during the whole sequence in a fixed dimension vector. For example, Metallinou et al. (2013) used a set of feature functions to summarise dialogue history using sums, averages, and other statistics (e.g. the number of times a hypothesis has appeared at a certain rank or the accumulated SLU confidence score for a certain value). Another option it to use a sliding window of length N (Henderson et al., 2013), conditioning the state tracking on the last N dialogue turns.

Sequential state trackers, in contrast to static classifiers, directly model the sequential nature of the problem. Linear-chain Conditional Random Fields have been used to model arbitrary length dialogues (Kim and Banchs, 2014, Lee and Eskenazi, 2013). However these models must design discrete feature functions, so continuous features (such as confidence scores) must be quantised. DST models based on RNNs, in contrast, have been shown to be able to deal with high dimensional continuous input features (Henderson et al., 2014c,d). They have also been shown to have a good performance operating directly on the N-best list output of the ASR, without requiring an SLU system[1]. This has two key benefits: firstly, it removes the need for hand-crafted feature design. And secondly, it avoids the engineering task of building a separate SLU model. In the next section, RNN based sequential discriminative dialogue state trackers are reviewed in more detail.

[1]This could be seen and modelling jointly the SLU and DST modules.

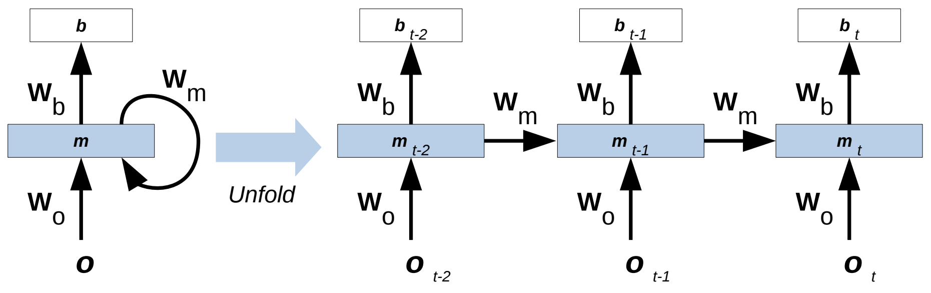

Figure 2.5: *Basic RNN architecture and an RNN unfolded through time.* **W** *represents the parameters of the model (weights),* **o** *the input,* **m** *the memory layer or hidden layer and* **b** *the output of the network. t denotes the time-step.*

2.5.3 Recurrent Neural Network based DST

An RNN is a class of artificial neural network (Bengio, 2009, Bishop, 2006) in which connections between units form a directed cycle (Elman, 1990, Jordan, 1997, Schmidhuber, 2015), letting them make use of sequential information. RNNs have an internal state (sometimes called "memory") which encodes information about the sequence of inputs seen so far, allowing them to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs, making them suitable for DST.

Figure 2.5 shows the basic structure of an RNN. The right part of the figure shows an RNN "unfolded" (or unrolled) through time, which can be seen as a large feedforward neural network in which the parameters of the sub-networks are shared across all time steps. Unfolding simply means writing the network for the complete sequence. The basic formulas that govern the computation in an RNN are as follows:

$$\begin{aligned} \mathbf{m}_t &= f(\mathbf{W}_m \mathbf{m}_{t-1} + \mathbf{W}_o \mathbf{o}_t) \\ \mathbf{b}_t &= \text{softmax}(\mathbf{W}_b \mathbf{m}_t + \mathbf{d}_b) \end{aligned} \tag{2.6}$$

where $\mathbf{o}_t$ is the input at time step t (e.g. the N-best output of the SLU plus the last action taken by the system), $\mathbf{m}_t$ is the hidden layer or *memory*, and $\mathbf{b}_t$ is the output of the network in each time step (the belief state in each turn in DST). $\mathbf{W}$ and $\mathbf{d}$ are the weight matrices and biases of each connection in the RNN. The function f is usually a non-linearity such as *tanh* or any *logistic function*.

Note that the hidden layer $\mathbf{m}_t$ captures information about what happened in all the previous time steps by combining its previous value $\mathbf{m}_{t-1}$ with the current input $\mathbf{o}_t$. Then, the output at step $\mathbf{b}_t$ is calculated solely based on the memory at time t. This is similar to the approach taken by generative state tracking approaches (e.g. equation 2.5) which computed the belief state $\mathbf{b}_t$ based on the previous one $\mathbf{b}_{t-1}$ and the current input $\mathbf{o}_t = (\omega_t, a_t)$. RNNs, however, discriminatively encode the dialogue history in a "latent" dialogue state and then use a feedforward layer to transform this encoding into the actual belief state.

RNNs are trained in a similar way to traditional Neural Networks. A modification of the *backpropagation algorithm*, called Backpropagation Through Time (BPTT) is used (Werbos, 1988). However, RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between turns that are far apart) due to what is called the *vanishing/exploding gradient* problem (Bengio et al., 1994). Certain types of RNNs (notably Long-Short Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997)) implement special layers and gating mechanisms to deal with this problem.

General dialogue state tracking

In the first two DSTCs, most of the data driven approaches to dialogue state tracking learned specific statistics for each slot and value (Bohus and Rudnicky, 2006, Williams, 2014). In slot-based dialogue systems, however, the set of possible values for each slot $\mathcal{V}_s$ can be very large and some slots may have values that occur very infrequently (e.g. food-type in the second DSTC). If a discriminative state tracker models the possible values for each slot as the different output classes, the model would have problems tracking values seen very infrequently or not seen at all in the training data. This will also occur when extending the domain of a dialogue state tracker. Some models addressed this issue by using parameter tying across slot models (Henderson et al., 2014d, Lee, 2013), assuming that the statistics of two slots are similar. In the third DSTC, the problem of domain extension was addressed and therefore state trackers able to generalise to unseen dialogue states had to be developed.

One of the most successful approaches to general state tracking (Henderson et al., 2014c) combined the output of two RNNs: one modelling slot-specific statistics and the other modelling slot-value independent general statistics. The slot-value independent statistics were modelled using *value specific general features* (also

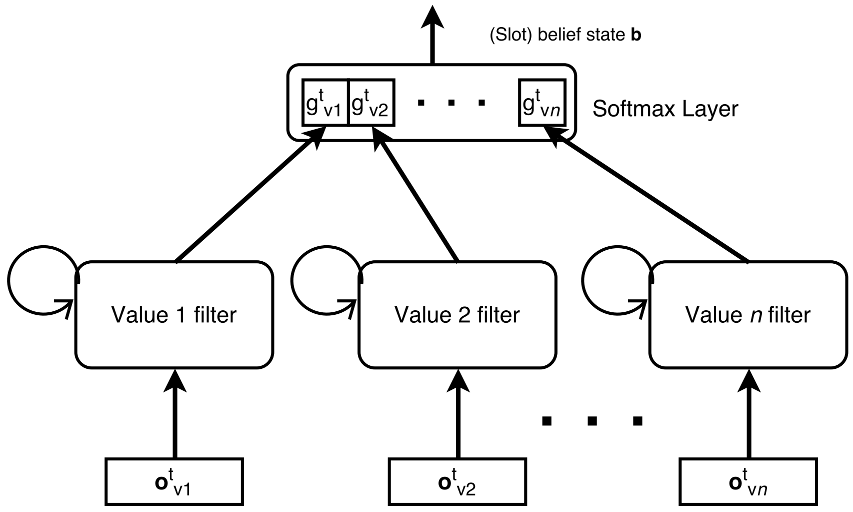

Figure 2.6: *General Dialogue State Tracking for a single slot. Each filter is an independent sequential classifier (e.g. an RNN) associated with a value v.* $\mathbf{o}_v^t$ *represent the value-specific input features at time-step t.* g_v^t *is the output of filter v.*

called *delexicalized features*), features present in each value extracted independently for each value (e.g. the confidence score of that value in the SLU output). Later, Mrkšić et al. (2015) modified this model to be able to track the dialogue state in completely different domains by using only the general part of the model of Henderson et al. (2014c). This slot-value independent model (shown in Fig. 2.6) comprises of a set of binary classifiers or *value filters*[1], one for each slot-value pair, with parameters shared across all filters. These filters track each value independently using the value specific general features. Each slot output distribution in each turn t is obtained by concatenating the outputs of each value filter g_v^t in $\mathcal{V}_s$, followed by applying a *softmax* function.

The set of filters differ from each other in two aspects: the input composed by value specific general features $\mathbf{o}_v^t$; and the label used during the training. An RNN-based general state tracker[2] updates the probability (or belief) of each value b_v^t in each turn t as follows:

$$\begin{aligned}
\mathbf{m}_v^t &= \sigma(\mathbf{W}_o\mathbf{o}_v^t + \mathbf{W}_m\mathbf{m}_v^{t-1} + \mathbf{d}_m)\\
g_v^t &= \sigma(\mathbf{w}_g\mathbf{m}_v^t + d_g)\\
b_v^t &= \frac{\exp(g_v^t)}{\sum_{v'\in V}\exp(g_{v'}^t)}
\end{aligned} \tag{2.7}$$

Where $\mathbf{m}_v^t$ is the hidden state of each filter and $\mathbf{W}_o$, $\mathbf{W}_m$, $\mathbf{d}_m$, $\mathbf{w}_g$ and d_g are the

[1]Addressed as *filters* due to their resemblance with convolutional neural networks filters.
[2]This is a simplified version of the model described in (Mrkšić et al., 2015).

parameters of the model. In summary, these models learn to track each value independently by learning a value tracking general model, and then combine the outputs of the independently tracked values. If the model needs to be extended to track a new value, an extra value filter for the new value would be added to the model. Therefore, these models work under the assumption that the state tracking statistics for each value are similar.

2.6 Dialogue Policy Optimization

Once the dialogue state or belief state has been estimated in each turn, the dialogue manager's next step is to choose which is the best action to take. In a MDP model[1], decisions on which action to take are determined by a policy, $\pi(s) = a$, which is a mapping from the states of the system to the possible actions. The aim of reinforcement learning is to obtain the optimal policy, i.e. the policy that maximises the long-term accumulated reward. In episodic tasks such as DM, the accumulated reward, R_t^π, for a particular policy π is the sum of the discounted immediate rewards that the policy π achieves from time t until the final time step T:

$$R_t^\pi = \sum_{i=0}^{T} \gamma^i r_{t+i+1} \tag{2.8}$$

If the state transitions are random and/or the immediate reward is a random process, then the accumulated reward is also a random process.

RL approaches often model the *value function* $V^\pi(s)$ for each state $s \in \mathcal{S}$; the expected accumulated reward R_t^π in state s when a system follows policy π:

$$V^\pi(s) = E_\pi(R_t^\pi | s_t = s) = E_\pi\Big(\sum_{i=0}^{T} \gamma^i r_{t+i+1} | s_t = s\Big), \tag{2.9}$$

where the expectation E_π is calculated over all possible state sequences that can be generated with policy π. In a similar way, the *Q-function*, $Q^\pi(s, a)$, for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ is also modelled; the expected discounted return that is obtained when action a is taken in state s and the policy π is followed from then on:

$$Q^\pi(s,a) = E_\pi(R_t^\pi | s_t = s, a_t = a) = E_\pi\Big(\sum_{i=0}^{T} \gamma^i r_{t+i+1} | s_t = s, a_t = a\Big), \tag{2.10}$$

[1]For simplicity, some of the policy optimization methods will be explained in the context of MDPs. However, as POMDPs can be seen as continuous state MDPs, the methods can be easily applicable to POMDPs.

where, again, the expectation E_π is calculated over all possible state sequences that can be generated with policy π.

The aim of RL-based PO is to obtain the optimal policy – i.e., the policy that maximises the value function. Assuming a finite state space $\mathcal{S}$, the exact solution to the optimal value function is given by the *Bellman optimality equation*: (Bellman, 1956):

$$V(s) = \max_a \sum_{s' \in \mathcal{S}} T(s,a,s')(R(s,a) + \gamma V(s')). \tag{2.11}$$

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

$$Q(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s')(R(s,a) + \gamma \max_{a'} Q(s',a')). \tag{2.12}$$

The optimal Q-function $Q(s,a)$ and the optimal value function $V(s)$ are related by equality

$$V(s) = \max_a Q(s,a). \tag{2.13}$$

Then, the optimal policy can be derived either from the optimal value function

$$\pi(s) = \arg\max_a \sum_{s'} T(s,a,s')(R(a,s) + \gamma V(s')), \tag{2.14}$$

or from the optimal Q-function

$$\pi(s) = \arg\max_a Q(s,a), \tag{2.15}$$

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

POMDP models, however, do not know the true state s. Therefore, the POMDP policy, $\pi(\mathbf{b}) = a$, must be a function of the belief state, providing an action $a \in \mathcal{A}$ for every possible belief state. Even if the true state is not known, the optimal value function for each state can be still computed using the observation function $O(\omega,s,a)$ and the observations $\omega \in \Omega$ seen (Kaelbling et al., 1998) as:

$$V(s) = \max_a \sum_{s' \in \mathcal{S}} T(s,a,s')\Big(R(s,a) + \sum_{\omega \in \Omega} O(\omega,s,a)\gamma V(s')\Big) \tag{2.16}$$

In a similar way, the optimal Q-function for POMDPs can be computed as:

$$Q(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s')\Big(R(s,a) + \max_{a'} \sum_{\omega \in \Omega} O(\omega,s,a)\gamma Q(s',a')\Big) \tag{2.17}$$

Then, the optimal value function for any possible belief state $V(\mathbf{b})$ can be computed as the weighted sum over all states using equations 2.6

$$V(\mathbf{b}) = \sum_{\mathbf{s} \in \mathcal{S}} b(s_t = s) V(s) \tag{2.18}$$

Where $b(s)$ is the value of the belief state for the state s. In a similar way, the optimal Q-function can also be computed as a weighted sum using equation 2.17:

$$Q(\mathbf{b}, a) = \sum_{\mathbf{s} \in \mathcal{S}} b(s_t = s) Q(s, a) \tag{2.19}$$

In order get the optimal policy by solving equations 2.18 or 2.19 directly, several assumptions are needed. Firstly, as mentioned before, the Markov property has to be satisfied. It can be difficult to design a dialogue state set that satisfies the Markov property while being simple enough to make learning possible. Secondly, the dynamics of the environment need to be known (the transition, observation and reward functions). Even if the dialogue state can be approximated, the reward function can be heuristically defined and the dynamics can be hand-crafted or estimated from data, obtaining the POMDP solution directly from the Bellman equation is intractable for large state spaces (Sutton and Barto, 1998).

RL literature proposes two main groups of approaches to compute the (possibly approximated) policy: *Model-based* and *model-free* approaches. Model-based approaches directly model the POMDP transition and observation functions, possibly approximating them using other sub-models (Kaelbling et al., 1998). These are very structured models which try to maintain the original POMDP structure. Model free approaches, on the contrary, do not make any assumption about the POMDP structure, modelling the Q-function directly from the observed rewards (Peters and Schaal, 2008). This gives the model more freedom to learn any underlying structure by itself. When applied to real world sized dialogue systems, however, existing model-based RL approaches become intractable (Williams and Young, 2007), and model-free approaches often need an extremely large number of dialogues to converge to the optimal policy (Jurčíček et al., 2010).

Algorithms to learn the optimal POMDP policy can also be divided into off-policy and on-policy methods. Off-policy methods estimate the optimal policy from examples obtained following another policy (which may be suboptimal). This can be useful when there is a corpus of machine-environment interactions available, obtained with another policy (or with another system) (Daubigney et al., 2012, Silver et al., 2016). To learn the optimal policy, however, every state action pair needs to appear in the corpus. On-policy methods, on the other hand, use the current best estimate of the optimal policy to choose the best action while reestimating the policy at the same time; they can learn on-line by directly interacting with the environment. On-policy methods often use an *exploitation-exploration* approach: at each time step, they choose to do *exploitation* or *exploration*. If they choose the former, they will take the optimal action based on their current policy. If they choose the latter, they will take an action that might be sub-optimal based on the current policy, but that will help to gather information about the environment to better estimate the true optimal policy. Using this approach new, and possibly better, strategies which would not occur following the original policy can be learnt.

In the next sections, the most popular model-based and model free algorithms to solve POMDPs in dialogue management are reviewed.

2.6.1 Model-based Policy Optimization

Model-based methods assume that the dynamics of the environment (i.e. the transition, observation and reward functions) are known. These dynamics can be either hand-crafted or learned from interactions with the environment. Dynamic programming (Bellman, 1956) (solving equation 2.14) is an example of a model-based method, but solving equation 2.14 directly is usually intractable. This equation, however, can be solved iteratively with the *value iteration* algorithm (Sutton and Barto, 1998). This algorithm makes use of the recursive property of the Bellman equation (equation 2.11) to relate optimal value function estimates between two consecutive time steps. If *1-step* denotes the time step before the final state, then the value function in *1-step* for some policy π in state s is the immediate reward $R(s, a)$, where action a is determined by the policy $\pi(s) = a$. For any time step (*t-step*), the value function of policy π in state s is the reward obtained by taking action a plus the discounted value function in *(t-1)-step* of policy π in the next state s'. Due to this recursive relationship and the stationary nature of the policy[1], it is possible to initialise the value function arbitrarily for *1-step* and iterate to update and maximise the value function for each subsequent step. This process is repeated until the difference between the *t-step* and the *(t-1)-step* value functions falls below a given threshold Δ. Algorithm 2.1 describes the process in detail.

The POMDP value function, which can be shown to be piecewise linear and convex (Cassandra et al., 1994), can also be obtained using the value iteration algorithm (Kaelbling et al., 1998). In this case, the value function is composed by a set of hyperplanes of dimension $|S| - 1$, where the upper surface of the intersection of these hyperplanes represents the optimal value function. Each hyperplane has

[1]The policy $\pi(s) = a$ is independent of the time step t.

Algorithm 2.1 Value iteration

```
1: $\Delta \leftarrow \infty, \theta \leftarrow$ arbitrary
2: for all $s \in \mathcal{S}$ do
3:     $V(s) \leftarrow$ arbitrary
4: end for
5: while $\Delta > \theta$ do
6:     for all $s \in \mathcal{S}$ do
7:         $v \leftarrow V(s)$
8:         $V(s) \leftarrow \max_a \sum_{s' \in \mathcal{S}} T(s,a,s')[R(s,a) + \gamma V(s')]$
9:         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
10:    end for
11: end while
12: for all $s \in \mathcal{S}$ do
13:     $\pi(s) = \max_a \sum_{s' \in \mathcal{S}} T(s,a,s')[R(s,a) + \gamma V(s')]$
14: end for
```

an action associated to it. Instead of the single update done in each iteration in the MDP case (line 7 in algorithm 2.1), each iteration of the Value iteration algorithm for POMDPs consists of two parts: a generation step – updating the value function at step $t+1$ using the hyperplanes on step t, and a pruning step – removing the hyperplanes that fall under the upper surface of the intersection. This process is repeated until there is no change in the optimal value function estimation for two consecutive steps. However, every iteration of the value iteration algorithm for POMDPs has exponential complexity which makes it intractable even for small scale problems. Approximate solutions such as in the *Point-based value iteration* algorithm (Pineau et al., 2003) exist, but these are still only suitable for relatively small action/state space problems.

2.6.2 Model free policy optimization

Model-free methods do not make any assumption about the underlying model of the environment, directly learning the Q-function online through interaction with the environment. These approaches often use an *ϵ-greedy* learning approach, selecting a random action to explore the state space with probability ϵ, or taking the action according to the best current policy with probability $1-\epsilon$:

$$\pi^e(s) = \begin{cases} \arg\max_{a \in \mathcal{A}} Q^\pi(s,a) \text{ with prob. } (1-\epsilon) \\ \text{random } a \in \mathcal{A} \text{ with prob. } \epsilon \end{cases} \tag{2.20}$$

Monte-Carlo (MC) methods and *Temporal Difference (TD) learning* are the most common model-free methods (Sutton and Barto, 1998).

Algorithm 2.2 Monte-Carlo control

```
1: for all $s \in \mathcal{S}, a \in \mathcal{A}$ do
2:     $Q(s,a) \leftarrow$ arbitrary
3:     $N(s,a) \leftarrow$ arbitrary
4:     $\pi(s) \leftarrow$ arbitrary
5: end for
6: while not convergence do
7:     Generate an episode with policy $\pi$
8:     for all $(s,a)$ pairs appearing in the episode do
9:         $R \leftarrow$ discounted return following the first occurrence of $(s,a)$
10:        $Q(s,a) \leftarrow \frac{Q(s,a)N(s,a)+R}{N(s,a)+1}$
11:        $N(s,a) \leftarrow N(s,a)+1$
12:    end for
13:    for all $s$ in the episode do
14:        $\pi(s) = \arg\max_a Q(s,a)$
15:    end for
16: end while
```

Episodic tasks (e.g. dialogue) define a start state and a collection of terminal states. An episode is then defined as a sequence of states from the initial to a terminal state, generated by interaction with the environment. MC methods, such as the Monte Carlo control algorithm (algorithm 2.2), are the most straight forward model free approach for episodic tasks. They directly estimate the Q-function from the observed accumulated rewards seen from each belief-action pair after each episode. The Monte Carlo control algorithm starts by taking actions according to an arbitrary policy and generating episodes, recording the sequence of states visited and actions taken in each episode. At the end of each episode, the accumulated reward from each state-action pair is computed and used to update the corresponding $Q(s,a)$ value. A count of the total number of times, $N(s,a)$, that each action pair has been visited is maintained to compute the weight of each $Q(s,a)$ update. In summary, the policy is updated after each episode to select the actions that have accumulated the highest reward for each state. This algorithm fits particularly well in PO of dialogue managers, where each dialogue is an episode and the accumulated reward can be given by the user or an independent evaluator at the end of the dialogue.

Monte-Carlo methods, however, cannot identify which particular state-action pairs had more influence in the outcome of an episode. Therefore, a larger number of interactions are needed to converge to the optimal policy (Sutton and Barto, 1998). TD learning addresses this issue by combining the ideas of MC and dynamic programming methods. As with Monte Carlo, learning is performed from

Algorithm 2.3 Sarsa

```
1: for all $s \in \mathcal{S}$,$a \in \mathcal{A}$ do
2:     $Q(s,a) \leftarrow$ arbitrary
3: end for
4: for all episode do
5:     Initialise $s$
6:     Choose $a$ from $s$ using policy derived from $Q$
7:     for all step in the episode do
8:         Take action $a$, observe $r$, $s'$
9:         Choose $a$ from $s$ using policy derived from $Q$
10:        $Q(s,a) \leftarrow Q(s,a) + \alpha[\gamma Q(s',a') - (Q(s,a) - r)]$
11:        $s \leftarrow s'; a \leftarrow a'$
12:    end for
13: end for
```

experience, but the Q-value estimates are updated after each step, using the difference between the obtained and expected rewards – the *temporal difference*. This entails a recursive relation similar to the one used in dynamic programming algorithms. *Sarsa* (algorithm 2.3) is the most popular on-policy TD algorithm, where the temporal difference is computed as the difference between the discounted future Q-value estimate, $\gamma Q(s',a')$, and the current Q-value estimate minus the observed immediate reward, $Q(s,a) - r$ (see line 10 in algorithm 2.3).

Even if this algorithm converges faster to the optimal policy than MC methods, the state-action space must be fully explored in order to optimise the policy. In real world dialogue systems, this space can be very large and exploring it with *ϵ-greedy* policies can be very inefficient. In addition, when this algorithm is applied to POMDPs models, the belief state must be quantised. A variation of *Sarsa* based on Gaussian Processes (GPs), named *GP-Sarsa* (Engel et al., 2005) solves this issues: it can directly work in the belief-action space (avoiding quantisation) and computes the uncertainty of the Q-value prediction for every belief-action pair, which can lead to a policy that explores more efficiently.

Gaussian Process Reinforcement Learning based Policy Optimization

A GP is a statistical model in which every point in some continuous input space is associated with a normally distributed random variable (Rasmussen, 2006). Moreover, given a finite collection of n random variables $\mathbf{X}_n$ (e.g. data points), a joint multivariate normal distribution of dimension n can be defined. A GP, denoted as $f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}'))$, is fully specified by a covariance or *kernel* function $k(\mathbf{x},\mathbf{x}')$ and a mean function $m(\mathbf{x})$. The kernel function is a positive semidefinite function that specifies correlations between the random variables $\mathbf{x}$ and $\mathbf{x}'$

(Schölkopf and Smola, 2002). A GP regression model (Rasmussen, 2006) tries to approximate the unknown function $f(\mathbf{x}_*) = y_*$ for any new point $\mathbf{x}_*$, given a set of observed data points $\mathbf{X}_n$ with corresponding target values $\mathbf{y}_n$. To do so, a zero-mean Gaussian process is usually defined, $f(\mathbf{x}) \sim GP((0), k(\mathbf{x}, \mathbf{x}'))$. If the observed points $\mathbf{X}_n$ are noisy, however, the true value of the function is unknown and assumed to be $y_i = f(\mathbf{x}_i) + \Sigma$, where the noise is additive, independent and Gaussian distributed, $\Sigma \sim \mathcal{N}(0, \sigma^2)$. As the model is assumed to follow a joint multivariate normal distribution, the joint distribution of the training outputs $\mathbf{y}_n$ and the test output $f(x_*)$ for any new test input x_* can be written as:

$$\begin{bmatrix} \mathbf{y}_n \\ f(x_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{X,X} + \sigma^2 \mathbf{I}_n & \mathbf{K}_{X,*} \\ \mathbf{K}_{*,X} & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right) \tag{2.21}$$

where $\mathbf{K}_{X,X}$ is the covariance matrix or *Gram* matrix computed with the kernel function between each pair of data points in $\mathbf{X}_n$,

$$\mathbf{K}_{X,X} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{2.22}$$

and $\mathbf{K}_{*,X}$ and $\mathbf{K}_{X,*}$ are the covariance vectors computed with the kernel function between the new data point $\mathbf{x}_*$ and the previously observed ones,

$$\mathbf{K}_{*,X} = \mathbf{K}_{X,*}^{\top} = \left[k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \dots, k(\mathbf{x}_n, \mathbf{x}_*) \right] \tag{2.23}$$

Then, conditioning the joint Gaussian prior distribution in equation 2.21 on the observations $\mathbf{X}_n$ and $\mathbf{y}_n$, the posterior distribution for any new point in the input space $\mathbf{x}_*$ can be computed as:

$$\begin{aligned} &f(\mathbf{x}_*) | \mathbf{X}_n, \mathbf{y}_n \sim \mathcal{N}(\bar{f}(\mathbf{x}_*), \hat{f}(\mathbf{x}_*)) \\ &\bar{f}(\mathbf{x}_*) = \mathbf{K}_{*,X} (\mathbf{K}_{X,X} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{y}_n \\ &\hat{f}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_{*,X} (\mathbf{K}_{X,X} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{K}_{X,*} \end{aligned} \tag{2.24}$$

where $\bar{f}(\mathbf{x}_*)$ and $\hat{f}(\mathbf{x}_*)$ represent the mean and variance of a single dimensional normal distribution respectively. A more detailed explanation of the derivation of equation 2.24 is given in (Rasmussen, 2006).

Recently, Gaussian Process based Reinforcement Learning (GP-RL) (Engel, 2005, Engel et al., 2003, 2005) has been proposed for dialogue policy optimization in the form of the *episodic GP-Sarsa* algorithm (Gašić and Young, 2014); a model-free[1], TD-based algorithm. This method can work in continuous space MDPs, thus avoiding the need for discretising the belief space in POMDPs (Gašić, 2011). It also computes the uncertainty of the Q-value estimate, which can be used as a metric for active learning during RL exploration to speed up the policy learning (Gašić and Young, 2014, Geist and Pietquin, 2010). These two properties can reduce the number of interactions needed to converge to the optimal policy by an order of magnitude with respect to other POMDP dialogue models, allowing to learn the policy directly from interacting with real users (Gašić et al., 2011). In addition, using transfer learning methods (Taylor and Stone, 2009) to initialise the policy with data gathered from dialogue systems in different domains has been shown to increase the learning speed of the policy further (Gašić et al., 2013), even providing an acceptable system performance when domain specific data was not available. As a non-parametric method, GP-RL is especially appropriate for systems where the amount of dialogue data begins small but increases as the user interacts with the system, such as homeService (section 2.1.1).

Recalling equation 2.10, the Q-function defines the expected accumulated reward when the dialogue is in belief state $\mathbf{b}_i$ and action a_i is taken, following policy π. GP-RL can be used to model the value of the Q-function as single dimensional Gaussian distribution given a set of size t of previously observed belief-action points $\mathbf{X}_t$ (the belief-action pairs seen so far in all previous dialogues), with their respective immediate rewards $\mathbf{r}_t$. The mean of this distribution will represent the expected value of Q and the variance the uncertainty of the expectation. The following paragraphs briefly explain the modelling of Q as a GP-RL model[2].

Each accumulated reward c_i observed at time-step[3] i corresponding to the belief-action point $\mathbf{x}_i = (\mathbf{b}_i, a_i)$ is a random variable. These random variables can be modelled as the sum of a mean Q plus a residual ΔQ:

$$c_i = Q(\mathbf{b}_i, a_i) + \Delta Q(\mathbf{b}_i, a_i) \tag{2.25}$$

Taking a TD approach, equation 2.10 and equation 2.25 can be combined to write the immediate reward r_i recursively as the temporal difference between the mean Q plus the residual at time i and at time $i+1$:

[1]Model-based GP-RL approaches do also exist (Deisenroth et al., 2009, Rasmussen et al., 2003), but have not been applied to dialogue.

[2]For a more detailed explanation the reader should refer to (Engel, 2005).

[3]In the following sections, the index i (and j) is used to refer to any time-step, while the index t is used to refer to the last time step of the set. When used as the subindex of matrices or vectors, t refers to the size of the matrix or vector.

$$r_i = Q(\mathbf{b}_i, a_i) + \Delta Q(\mathbf{b}_i, a_i) - \gamma_i Q(\mathbf{b}_{i+1}, a_{i+1}) - \gamma_i \Delta Q(\mathbf{b}_{i+1}, a_{i+1}) \tag{2.26}$$

where, in episodic tasks, $\gamma_i = 0$ if a_i is a terminal action, and a normal discount factor $1 \geq \gamma \geq 0$ otherwise. Given the set of observed belief-action points $\mathbf{X}_t$, with their respective immediate rewards $\mathbf{r}_t$, the set of linear equations arising from equation 2.6.2 can be represented in matrix form as:

$$\mathbf{r}_{t-1} = \mathbf{H}_t \mathbf{q}_t + \mathbf{H}_t \Delta \mathbf{q}_t \tag{2.27}$$

where:

$$\mathbf{r}_{t-1} = [r_1, r_2, ..., r_{t-1}]^\top$$
$$\mathbf{q}_t = [Q(\mathbf{b}_1, a_1), ..., Q(\mathbf{b}_t, a_t)]^\top$$
$$\Delta\mathbf{q}_t = [\Delta Q(\mathbf{b}_1, a_1), ..., \Delta Q(\mathbf{b}_t, a_t)]^\top$$
$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma_1 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -\gamma_{t-1} \end{bmatrix}$$

If the set of random variables $\mathbf{q}_t$ is assumed to have a joint Gaussian distribution with zero mean and the residuals ΔQ are assumed to be white independent noise, $\Delta Q(\mathbf{b}_i, a_i) \sim \mathcal{N}(0, \sigma^2) \forall i$, the set of equations 2.27 can be modelled as a zero-mean GP (Engel, 2005). To compute the covariance matrix of the GP, a *kernel function* must be defined in the belief-action space. This can be done by factorising the kernel function as a kernel in the belief space k^b and a kernel in the action space k^a (Engel et al., 2005):

$$k_{i,j} = k((\mathbf{b}_i, a_i), (\mathbf{b}_j, a_j)) = k^b(\mathbf{b}_i, \mathbf{b}_j) k^a(a_i, a_j) \tag{2.28}$$

In MDP models with discrete action spaces (such as dialogue), the kernel over the action space is usually defined as the delta kernel (Gašić et al., 2011):

$$k^a(a_i, a_j) = \delta(a_i, a_j) = \begin{cases} 1 \text{ if } a_i = a_j \\ 0 \text{ otherwise} \end{cases} \tag{2.29}$$

After the kernel function has been defined, the posterior distribution of the Q-function for any new belief action point $\mathbf{x}_* = (\mathbf{b}_*, a_*)$ given the set of previously observed points and immediate rewards can be computed as:

$$\begin{aligned} Q(\mathbf{x}_*)|\mathbf{X}_t, \mathbf{r}_{t-1} &\sim \mathcal{N}(\bar{Q}(\mathbf{x}_*), \hat{Q}(\mathbf{x}_*)) \\ \bar{Q}(\mathbf{x}_*) &= \mathbf{K}_{*,X}\mathbf{H}_t^\top(\mathbf{H}_t\mathbf{K}_{X,X}\mathbf{H}_t^\top + \mathbf{\Sigma}_t)^{-1}\mathbf{r}_{t-1} \\ \hat{Q}(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_{*,X}\mathbf{H}_t^\top(\mathbf{H}_t\mathbf{K}_{X,X}\mathbf{H}_t^\top + \mathbf{\Sigma}_t)^{-1}\mathbf{H}_t\mathbf{K}_{X,*} \end{aligned} \tag{2.30}$$

where $\mathbf{K}_{X,X}$ is the Gram matrix (equation 2.22) of the set of belief-action points $\mathbf{X}_t$ computed with the kernel in the belief-action space and $\mathbf{K}_{*,X}$ and $\mathbf{K}_{X,*}$ are the covariance vectors (equation 2.23) between $\mathbf{X}_t$ and $\mathbf{x}_*$. $\mathbf{\Sigma}_t = \sigma^2\mathbf{H}_t\mathbf{H}_t^\top$ is the additive noise term and $\bar{Q}$ and $\hat{Q}$ represent the mean and the variance of the Q-function, respectively.

Note how the difference of the GP-RL model (equation 2.30) with the GP regression model (equation 2.24) arises from the inclusion of the operator $\mathbf{H}_t$. This is because the GP-RL model is estimating the expected accumulated reward (Q-function) from the observed immediate rewards by using the recurrence defined by the *Bellman equation* (equation 2.11). The $\mathbf{H}_t$ operator defines the temporal difference relationship between two consecutive belief actions points.

Once the Q posterior for any new belief-action point can be computed using equation 2.30, the optimal policy $\pi^*(\mathbf{b}) = a$ can be computed as the action a that maximizes the Q-function from the current belief state $\mathbf{b}_*$, or using an ϵ*-greedy* policy (equation 2.20). GP-RL, however, computes the uncertainty of the expected accumulated reward in form of the variance $\hat{Q}$, which can be used as a metric for an *active exploration* policy π^a (Daubigney et al., 2011) to speed up the learning. This variation of the ϵ-greedy policy chooses the action with higher uncertainty in exploration steps, maximising the information about the environment obtained from taking that action:

$$\pi^a(\mathbf{b}_*) = \begin{cases} \arg\max_{a\in\mathcal{A}} \bar{Q}(\mathbf{b}_*, a) \text{ with prob. } (1-\epsilon) \\ \arg\max_{a\in\mathcal{A}} \hat{Q}(\mathbf{b}_*, a) \text{ with prob. } \epsilon \end{cases} \tag{2.31}$$

where ϵ controls the exploration rate.

2.7 Dialogue evaluation and user simulation

Contrary to other speech and natural language processing tasks where evaluation methods are well established (Hirschman and Thompson, 1997), evaluation of spoken dialogue systems is more difficult because it requires interaction. This difficulty arises from the fact that, contrary to most speech processing tasks, dialogue is a control problem: the actions taken by the user will depend on the actions taken by the system and vice-versa. Therefore, different dialogue managers will create different dialogue *trajectories*[1] while interacting with the same user in the exact same conditions. It is thus infeasible to evaluate the performance of the dialogue in a independent test set as is done in classification tasks such as ASR and SLU, as the corpus of test dialogues will be dependent on the dialogue manager it was collected with.

The direct approach to evaluate dialogues is to test the dialogue manager interacting with humans and let human judges (or the users themselves) rate the dialogues (Lemon et al., 2006, Singh et al., 2002). However, this approach has two major flaws: it is very costly and time-consuming, making it infeasible to evaluate all possible dialogues; and it depends on a subjective metric to evaluate the quality of the dialogue. The first problem can be tackled by creating a Simulated User (SU), a model of the user (or the whole environment) which can interact with the system in any number of dialogues (Schatzmann et al., 2006). The second problem can be tackled by using the RL accumulated reward (equation 2.8) as a measure of the quality of the dialogue (Levin et al., 2000). This requires, however, the definition of a good quality reward function. The next section reviews different approaches to reward function modelling and section 2.7.2 reviews different approaches to model SUs.

2.7.1 Reward functions

Reflecting the quality of the whole dialogue in a function depending on the immediate rewards $r_i = R(s_i, a_i)$ can be very challenging for some dialogue systems. In systems where the objective is to entertain the user, for example, it might be difficult to measure the "entertainment level" of the user in each turn. In task oriented dialogues, however, measuring the quality of the dialogue is much simpler. The objective of the dialogue can be summarised as performing the task that the user requests while spending the least time possible. The reward function can be then defined as a trade-off between maximising the dialogue success while minimising the dialogue length. A dialogue is considered successful when the system performs the task the user asked for, or when it informs that it cannot perform the task if the task is beyond the scope of the system (maybe providing alternatives). Correctly annotating a dialogue as successful, however, is not as trivial as it looks, and sometimes even the users label their own dialogues as successful when they are not (Gašić et al., 2011). Anyhow, a simple but effective reward function used

[1]A dialogue trajectory is defined as the sequence of actions, states, rewards and observations happening in a dialogue from the initial state until the terminal action.

in several studies (Casanueva et al., 2014, Jurčíček et al., 2010, Williams, 2007a, Young et al., 2010) gives a small negative reward each turn (to encourage short dialogues) and a large positive reward when the dialogue is completed successfully (to maximise the dialogue success rate). The ratio between these two values sets how much weight is given to prioritise short dialogues and how much to increase the success rate.

An alternative approach is to learn the reward function from ratings by human judges. The PARADISE evaluation framework (Walker et al., 1997) predicts the user satisfaction ratings using a corpus of dialogues annotated with real user satisfactions and a set of objective measures. This is similar to Inverse Reinforcement Learning (Abbeel and Ng, 2004, Ng et al., 2000), which learns the local reward function from a set of human-human dialogues annotated as successful or unsuccessful (El Asri et al., 2012). These approaches, however, still have the same flaw of supervised learning for DM: they need a large corpus with enough variability to learn a reward function that can work properly in any dialogue situation. Recent work has tried to learn to classify dialogues as successful or unsuccessful using a RNN based model (Su et al., 2015, 2016b), reducing the effort needed to annotate the dialogues.

2.7.2 Simulated Users

To evaluate a dialogue manager properly, a large amount of interactions with real users are required. If several different dialogue managers need to be compared, each one of them needs to be evaluated interacting with the same users for a large number of dialogues. This is extremely costly and infeasible in several situations, such as interacting with dysarthric speakers, for whom speaking during a prolonged period of time can be very tiring. In the case of online PO methods, the number of dialogues needed to converge to the optimal solution is often evaluated – the policy learning speed or rate. The amount of dialogues needed to train each policy model from scratch to evaluate the learning speed is infeasible for most studies.

A popular alternative to evaluation with real users is to build a SU, a model which tries to "mimic" the behaviour of real users interacting in a specific domain. With SUs, it is possible to generate the necessary number of dialogues. The main problem with SUs, however, is the potential discrepancy between the real and simulated behaviour. It is very difficult to model all the potential variability of real users[1], therefore, a good performance with a SU does not necessarily reflect

[1]technically, if the dynamics of the real users (and environment) are known, the dialogue management problem can be solved with model-based RL methods.

in a good performance with real users. SUs are, however, a good method to bootstrap data to train an initial dialogue policy and it is a common practice in RL-based dialogue management to use a SU to interact with the dialogue manager during policy optimisation (Jurčíček et al., 2012, Levin et al., 2000, Scheffler and Young, 2001, Thomson and Young, 2010). Interacting with the SU can also enable a wider coverage of dialogue space than interacting with human users. Early SU approaches only simulated the user behaviour (the utterance of the user in each turn) (Eckert et al., 1997), but modern approaches also simulate the ASR channel, being able to test the DM with varying ASR performance (Pietquin and Beaufort, 2005, Pietquin and Renals, 2002, Schatzmann et al., 2007b), or even the whole environment (the user and the ASR plus the SLU) (Thomson et al., 2012).

Simulated User behaviour

There are two key characteristics needed by SUs: reasonable, goal-directed behaviour and variability in the way it interacts with the dialogue manager so that it can cover a wide part of the dialogue space. Some approaches to user simulation use statistical models with hand-tuned parameters, for example graph-based (Scheffler and Young, 2001) and agenda-based (Schatzmann et al., 2007a) techniques. There are, however, several methods to learn the parameters of the user simulation model from a dialogue corpus, such as Bayesian networks (Pietquin and Dutoit, 2006), N-gram methods (Georgila et al., 2006), cluster-based user simulation (Rieser and Lemon, 2006) and agenda-based user simulation with parameters estimated from data (Keizer et al., 2010, Schatzmann and Young, 2009). However, these are all supervised learning methods, so the space of dialogues that can be generated will be restricted to the original space of the corpus it was trained on. To create SUs with wider coverage of the dialogue space than real users, mechanisms to increase the variability of the generated dialogues while maintaining the user behaviour coherence are necessary (Schatzmann, 2008, Schatzmann et al., 2006).

Simulated ASR and SLU

ASR and SLU simulation tries to model the "corruption" effect that these channels introduce in the true user utterance. It is usually modelled as a mapping from the real sentence uttered by the user to the output of the ASR (Pietquin and Renals, 2002), the 1-best output of the SLU (Schatzmann et al., 2007b), or an N-best list of SLU hypotheses (Thomson et al., 2012). Ideally, these models should be learnt from a corpus of dialogues with the true user sentence and the ASR or SLU output annotated, but the large variability of the input and output space makes this difficult. Several methods simplify the error simulation by just including a probability to randomly change some part of the output (Pietquin and Beaufort, 2005), which can be tuned to simulate different ASR or SLU performances.

2.7.3 Dialogue State Tracking evaluation

Even if the quality of a dialogue manager has to be evaluated measuring the overall outcome of the dialogue, evaluating the modules that compose an SDS or VUI independently can be useful – e.g. to compare different models. As DST is a supervised learning task, its performance can be evaluated in an independent test set of dialogues with annotated dialogue states, simplifying the comparison between different state trackers. The first DSTC (Williams et al., 2013) defined a large set of metrics useful to evaluate the DST performance, while the second and third challenges (Henderson et al., 2014a,b) identified the most useful ones: state tracking *accuracy* and *L2 measure*. State tracking accuracy computes the percentage of turns in which the top hypothesis of the output of the dialogue state tracker (the belief state) corresponds to the true dialogue state or user goal. *L2* measures the L^2 distance between the belief state and a vector of zeros with 1 in the position of the correct hypothesis, indicating the quality of the full probability distribution outputted by the tracker.

The DSTCs also defined two possible schemes to annotate the true dialogue state in each turn, named *scheme A* and *scheme B*. Under scheme A, each component of the state is defined as the most recently asserted value given by the user. The "None" value is used to indicate that a value has not been given yet. In *Scheme B*, labels are propagated backwards through the dialogue; the label at a current turn for the dialogue state is considered to be the next value that the user indicates. This labelling scheme is designed to assess whether a tracker is able to predict the users intention before it has been stated.

In addition, the DST challenges also define two *schedules* which are used to decide which turns to include when computing each metric. *Schedule 1* includes every turn, while *Schedule 2* only includes a turn if any SLU hypothesis up to and including the turn contains some information about the component of the dialogue state in question, or if the correct label of that slot is *None*. *Schedule 2* aims to evaluate only the dialogue turns in which the user has given enough evidence to know which the true user intention is.

Chapter 3

Statistical DM for personalised voice user interfaces

Voice User Interfaces (VUIs) are a very attractive alternative to conventional interfaces for people with severe physical disabilities who also suffer from dysarthria, but the low performance of the ASR poses a challenge when implementing these interfaces. Even if the adaptation of acoustic models to the user has been shown to improve the ASR accuracy for dysarthric speakers, the error rate is still high for users with moderate to severe dysarthria (Christensen et al., 2012a). POMDP based statistical DM has been shown to improve dialogue interaction performance in high ASR error rate conditions, making it a very attractive framework for VUIs developed for dysarthric speakers. Some small scale research has already been done on applying this framework to interfaces for speakers with mild to moderate dysarthria (Li et al., 2013).

One of the keys to improving the ASR performance with dysarthric speakers is the personalisation of the models to the specific speakers and systems. The acoustic models are adapted using speaker-specific acoustic data and the language model is restricted so the ASR recognises only the vocabulary of the specific task (Christensen et al., 2012a). Research in machine learning based DM, however, has usually focused on user-independent systems. In these systems, it is assumed that all the users have similar characteristics. Thus, a general model is trained using data coming from several speakers and this model is used by any speaker. Outside the work presented in this thesis, very few studies (and only very recently) have researched user adaptation for policy optimization (Chandramohan et al., 2014, Genevay and Laroche, 2016). User adaptation of dialogue management models, however, is a very promising area of research with many potential applications, such as personal assistants, smart homes and in-car control systems.

In addition, DM models based on machine learning usually assume a static environment. This means that the characteristics of the user, ASR and other modules of the environment do not change over time. However, if the ASR is adapted online with user-specific acoustic data gathered through interaction with the VUI, the environment dynamics will change over time and the DM will have to adapt to these changes. It has also been shown that a user who interacts with a system during a long period of time can adapt his/her pronunciation and behaviour to adapt to the system (Christensen et al., 2015), which from the point of view of the DM will be reflected as a change in the environment dynamics.

Nowadays, most assistive VUIs use rule-based dialogue managers (Christensen et al., 2013b, Vacher et al., 2011, 2015). However, statistical dialogue management in the form of POMDP-based models can be a better approach due to its greater adaptability to the characteristics of each speaker and to changes in the environment. In addition, the higher adaptability of POMDP dialogue managers can widen the range of the *optimal operating point* in a homeService (hS) system (see section 2.1.1) due to the ability of these models to adapt the policy to the ASR performance. If the ASR error rate is high, the policy will be more conservative – i.e. the number of grounding questions[1] and confirmations will be higher. Then, when the ASR error rate decreases, the policy will be more straight-forward. However, there are some important differences between the "typical" DM environment and a personalised VUI environment, which need to be taken into account to integrate POMDPs in personalised VUIs.

In this chapter, the characteristics of VUIs environments for assistive technologies are described, highlighting differences with the typical SDS environments. In addition, the performance improvement that can be obtained using model-based POMDP DM is explored, identifying the possible scenarios that can arise in personal VUIs for assistive technologies which are not seen in typical SDSs. In the first section, the "homeService environment" is presented and analysed, the personalised VUI environment in which all the experiments presented in this thesis are performed and evaluated. In section 3.2, different approaches to represent the dialogue state in a hS system are proposed and in section 3.3 a tractable model-based policy optimization is developed. Sections 3.4 presents experiments performed with the tractable model-based policy, analysing its performance in the special scenarios that might arise when interacting with personalised VUIs. The last section presents conclusions and reviews the POMDP-DM aspects that need to be adapted to personalised VUI scenarios, which will be researched in the next

[1]Grounding questions are any kind of questions used to disambiguate about any piece of information given by the user, such as confirmations or requests to repeat commands.

chapters.

3.1 The homeService environment

As explained in section 2.1.1, hS is a project where users with disabilities, who also suffer from speech disorders, are being provided with speech-driven environment control interfaces and interact with them in a longitudinal study. The users can interact with these VUIs by speaking single word commands, using them to control their home devices such as the lights or the TV. The screen of a tablet provides information about the state of these devices and shows the commands to control them. Figure 3.1 shows a picture of a hS system and its main components set up in a users home. In this project, the users effectively become part of the research team, discussing about the design and specifications of their personal system, such as the devices they want to control and the commands they want to utter to control them. The researchers work with the users to close the "virtuous circle", establishing an initial "operating point" for each user depending on their dysarthria severity: a VUI with a vocabulary small enough that a good performance can be expected from the ASR and yet sufficiently useful so that the user's interest is maintained. Working in an *optimal operating point* will keep the user engaged, letting the system collect more acoustic data and thus, letting the ASR acoustic model be adapted to improve its performance. Once the ASR performance is good enough, the vocabulary can be extended to let the VUI control more devices. The personalisation to each user (in the commands they use, the ASR acoustic models and to the dysarthria severity of the user) makes hS the perfect scenario to study the personalisation of DM models. In the next subsection, the architecture of a hS system from the point of view of the Reinforcement Learning (RL) framework is analysed.

3.1.1 hS from a Reinforcement Learning DM perspective

Figure 3.2 shows a schematic representation of the homeService system from an RL point of view. Following an RL approach, the dialogue manager acts as the RL *agent* which interacts with the RL *environment*, composed by the user plus the rest of the VUI components. The VUI components that are part of the environment can be clustered into two modules: the speech understanding system (the ASR) and the response generator.

Note that, in contrast to other typical SDSs, in hS there is no Spoken Language Understanding (SLU) system in the environment. For dysarthric speakers, it is difficult to articulate long sentences (Kim et al., 2008). Thus, the ASR is set up to

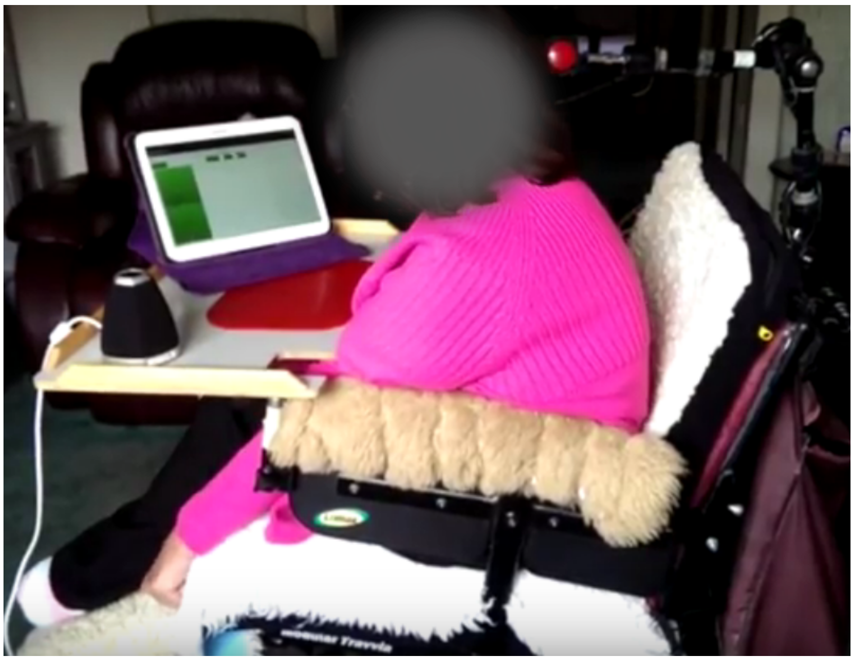

Figure 3.1: *Picture of a hS system set up in a users home (using rule based DM). The tablet gives the user feedback about the actual system state and shows the list of commands that he/she can utter.*

recognise only single word commands and to output an N-best list of words with confidence scores. This configuration improves the accuracy of the decoder as the space of classes to search from is constrained, making ASR usable for moderate and severe dysarthric speakers (Christensen et al., 2012a). It can be considered that the SLU is a one-to-one mapping from keywords appearing in the ASR output to SLU concepts, because the output of the ASR already lies in the space of concepts used as input by the state tracker.

The responses generated by the hS environment are also different to the usual responses of SDSs. The system includes a loudspeaker to output grounding questions if necessary, but most of the times the actions taken by the system will be commands to the home devices, which will be reflected in the tablet and in the actual home devices being controlled. The tablet acts as a personalised visual interface for the user, displaying a representation of the system state and the options available for the user (the commands that the system can take as input in each turn).

The dialogue manager (RL agent) is composed of a dialogue state tracker and a dialogue policy as in typical SDSs. As previously mentioned, the input to the state tracker in each dialogue turn is an N-best list of command hypotheses with associated confidence scores (the ASR output), plus the last action taken by the

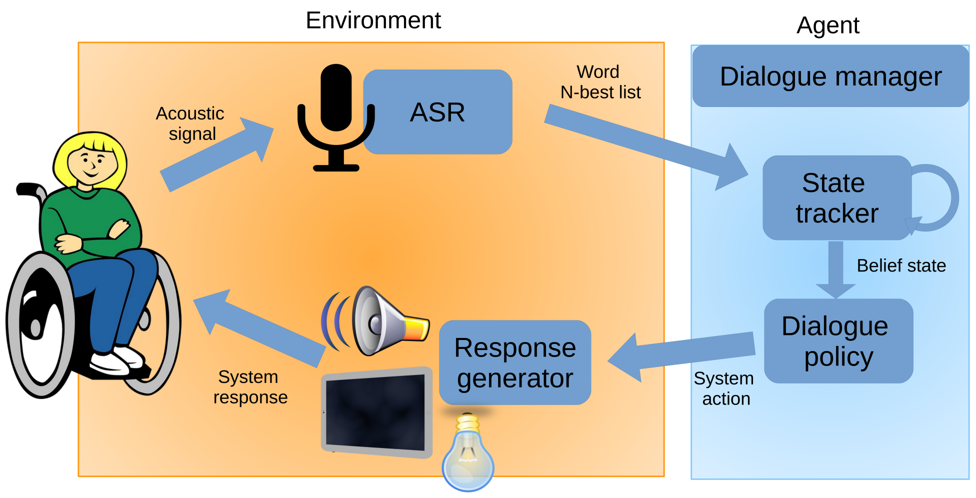

Figure 3.2: *Schematic representation of the homeService environment interacting with a Dialogue Manager (agent) from a Reinforcement Learning perspective. The orange box shows the modules belonging to the environment and the blue box the modules belonging to the agent.*

DM. The policy can output grounding questions through the loudspeaker, or actions that change the state of the devices through an infra-red emitter. Possible dialogue state representations for the dialogue manager are discussed in section 3.2.2.

3.1.2 Adaptive ASR for dysarthric speech

As explained in section 2.2.1, the most promising approach to make ASR usable by dysarthric speakers is to use speaker-specific data to adapt speaker independent acoustic models. In Christensen et al. (2012a), a comparison of different adaptation techniques is presented, showing that Maximum A Posteriori (MAP) adaptation (Gauvain and Lee, 1992) is one of the most promising approaches for dysarthric ASR. This study presents accuracy results on the *UASpeech* task (Kim et al., 2008), using the whole vocabulary (455 words) and about 40 minutes of acoustic data from each speaker to adapt the acoustic models. In a hS environment, however, the amount of speaker-specific data used to adapt the acoustic models varies, increasing as the user interacts with the system and more data is collected. In addition, the size of the command vocabulary of the system (the amount of words the

Speaker	Speech intelligibility	Dysarthria diagnosis	Data amount
M04	Very Low (2%)	Spastic	13.30
F03	Very Low (6%)	Spastic	19.11
M12	Very Low (7.4%)	Mixed	16.07
M01	Very Low (15%)	Spastic	10.0
M07	Low (28%)	Spastic	18.69
F02	Low (29%)	Spastic	18.84
M16	low (43%)	Spastic	16.19
M05	Mid (58%)	Spastic	18.42
F04	Mid (62%)	Athetoid (or mixed)	18.57
M11	Mid (62%)	Athetoid	16.38
M09	High (86%)	Spastic	18.42
M14	High (90.4%)	Spastic	19.34
M08	High (93%)	Spastic	18.38
M10	High (93%)	Undetermined	19.19
F05	High (95%)	Spastic	19.0
Mean Very Low	7.6%	Spastic	14.62
Mean Low	33.3%	Spastic	17.91
Mean Mid	60.6%	Athetoid	17.79
Mean High	91.5%	Spastic	18.87
Mean All	47.2%	Spastic	17.33

Table 3.1: *Statistics of each speaker in the UASpeech database. The intelligibility is the percentage of words that can be understood by unfamiliar speakers. The data amount shows the mean repetition count for each word by that speaker (excluding the uncommon words).*

language model is constrained to recognise) will affect the ASR performance too. In this section, the effect of having access to increasing amounts of speaker specific data to adapt the acoustic models with different vocabulary sizes is studied. The *UASpeech* database is used to perform these experiments.

UASpeech database

UASpeech (Kim et al., 2008) is by far the largest database of dysarthric speech suitable for training acoustic models for ASR. It contains speech recordings from 15 different speakers (4 female and 11 male). The recordings are composed by 5 groups of single words: 10 digits, 29 NATO alphabet letters, 19 command words (“back”, ‘shift”, etc.), 100 common words (“that”, “she”, etc.) and 300 uncommon words chosen to be phonetically rich. In total, the database contains around 70 minutes of speech for each speaker. A detailed description of the statistics of the database with the description of each speaker is presented in table 3.1 and full details of the corpora can be found in Kim et al. (2008). The database also contains some meta-data, such as the gender (first column, “M” or “F” in the speakers name indicates Male and Female, respectively) or the type of dysarthria of each speaker (third column). Another important meta-data annotation is a speaker intelligibility measure[1] (second column). These range from 4% to 95%. The speakers are clustered in four groups depending on their intelligibility measures (shown by different colors in the table); *very low*, (2% to 15%, 4 speakers, grey); *low*, (28% to 43%, 3 speakers, red); *mid*, (58% to 62%, 3 speakers, blue) and *high*, (86% to 95%, 5 speakers, green). The last column in the table shows the amount of data available per speaker, represented by the mean repetition count for each word in the vocabulary (excluding the uncommon words).

Effect of the amount of adaptation data and vocabulary size on ASR accuracy

MAP adaptation (Gauvain and Lee, 1992) has been shown to be a successful way of establishing acoustic models when faced with limited amounts of data from a given speaker. In Christensen et al. (2012a), MAP was shown to be a very promising approach to acoustic modelling for dysarthric speech, but it was only evaluated with fixed (large) amounts of adaptation data and fixed vocabulary sizes. In this section, a set of ASR experiments is performed using the *UASpeech* database to evaluate the effect of varying the amount of acoustic data used for MAP adaptation and the vocabulary size, using the same ASR configuration as Christensen et al. (2012a). The data is encoded in a 39 dimensional feature vector of PLP features with added first and second order time derivatives. The maximum likelihood criterion is used to train the HMMs. State-clustered triphones are used with Gaussian mixture models with 16 components per state. As the database consists of single word recordings, a uniform language model containing only the words in the database vocabulary is used. To study the performance with different vocabulary sizes, the amount of words in the language model is varied; 455 words (all vocabulary), 155 words (all except the uncommon words), 119 words (commands and uncommon words) and 36 words (digits and NATO vocabulary). For each of the 15 speakers in the database, a speaker independent model is trained using data from the other 14 speakers. These speaker independent models are adapted with different amounts of speaker specific data to simulate the effect of having access to increasing amounts of data collected through interaction with the user. The amount of data used to adapt the models is measured in “single word recordings”, varying for each vocabulary size.

The ASR accuracy results for these experiments are shown in figure 3.3. Four

[1]This measure is based on the percentage of correctly transcribed words obtained by five unfamiliar listeners.

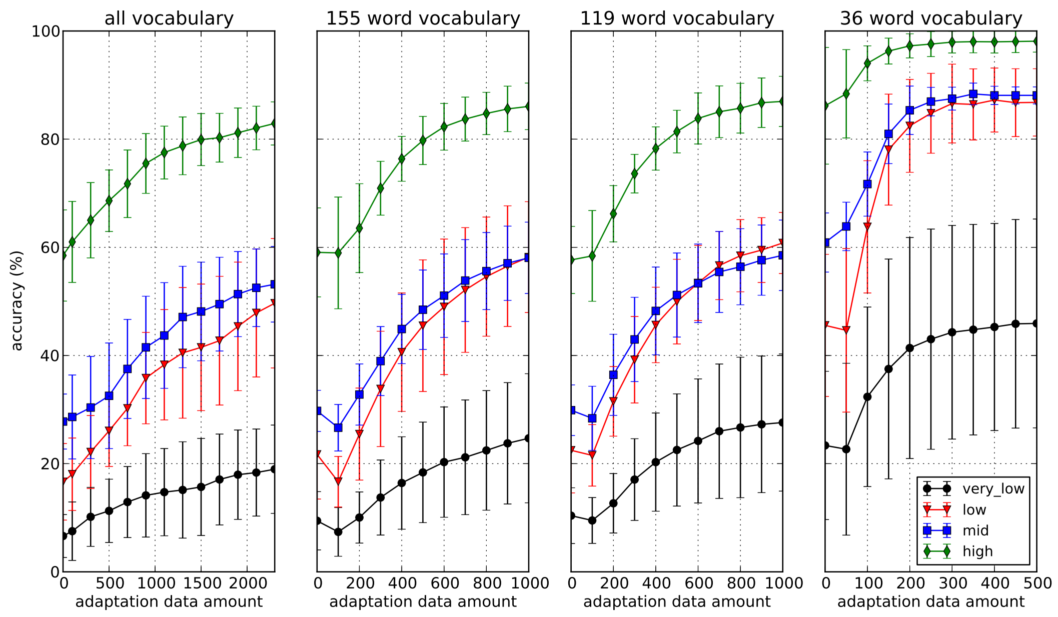

Figure 3.3: *Effect on accuracy of increasing the amount of data for MAP adaptation for different vocabulary sizes and for different intelligibility groups. The x axis represents the amount of word recordings used to adapt the acoustic models.*

different plots are presented, each of them showing the results with a different vocabulary size: 455, 155, 119, and 36. The y axis shows the ASR performance and the x axis the amount of single word recordings used to adapt the acoustic models (note that these amounts are different for each vocabulary size). The four lines plotted show the mean and standard deviation for each of the four intelligibility groups (Very Low, Low, Mid, and High). Analysing the results, three main observations can be made:

- The performance is highly dependent on the dysarthria severity (or intelligibility). The performance of the Low and Mid speakers is similar, so a further clustering into three groups can be made: Very Low intelligibility speakers, Low and Mid intelligibility speakers and High intelligibility speakers. The ASR accuracy for each of these groups is very different from the others, suggesting that a DM optimised for one group will not be optimal for the others. The high variance shown by the Very Low and Low-Mid groups also suggests that the DM should be personalised to each speaker.

- There is a strong correlation between the amount of acoustic data used for adaptation and the ASR accuracy. When the amount of speaker-specific data is small (Which would correspond to the initial usage stage of a hS system),

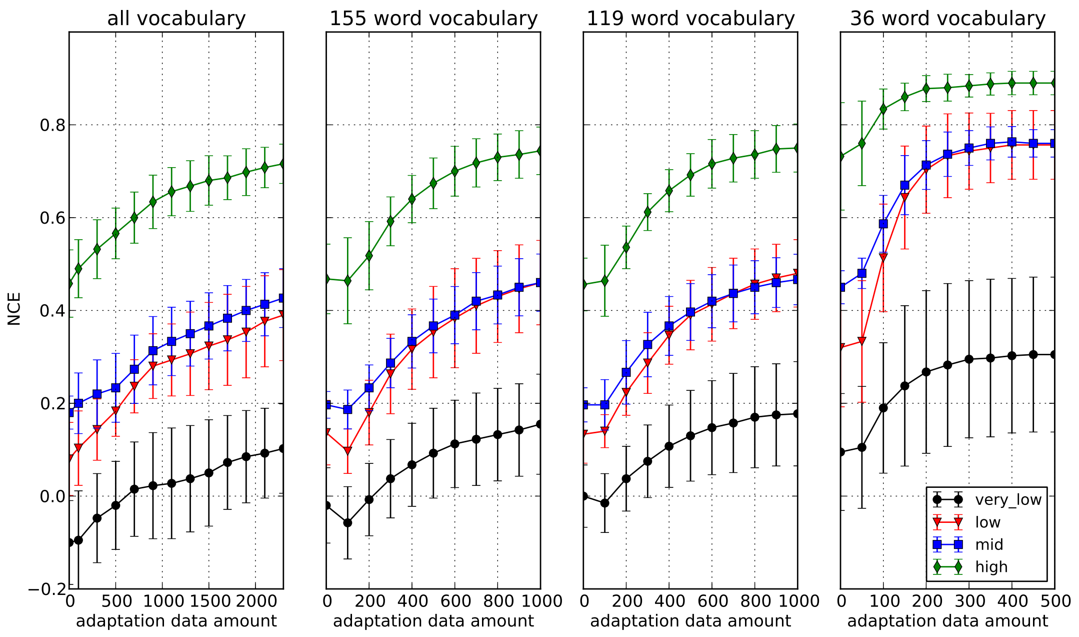

Figure 3.4: *Effect on NCE of increasing the amount of data for MAP adaptation for different vocabulary sizes and for different intelligibility groups. The x axis represents the amount of word recordings used to adapt the acoustic models.*

the relation is lineal, however, as the amount of data increases, the accuracy improvement begins to decrease until it converges. This suggests that the environment will be variable in the initial stages of the usage of a system, until there is enough data so the ASR accuracy converges.

- The performance is very dependent on the vocabulary size, not only on accuracy but also on the amount of acoustic data required to reach the convergence point.

In addition, the quality of the confidence scores is analysed. To do so, Normalised Cross Entropy (NCE) has been shown to be a good confidence score quality measure (Thomson et al., 2008). Figure 3.4 shows the NCE results for the same speakers and vocabularies presented in figure 3.3. It can bee seen that the NCE scores are very correlated with the ASR accuracy, with a Pearson correlation coefficient of 0.99. To visualise this correlation more clearly, figure 3.5 plots the NCE scores versus the accuracy for all the intelligibility groups and vocabularies. Therefore, the conclusions obtained from the figure showing the ASR accuracy can also be applied to the confidence scores.

The main conclusion that can be drawn from the experiments in this section

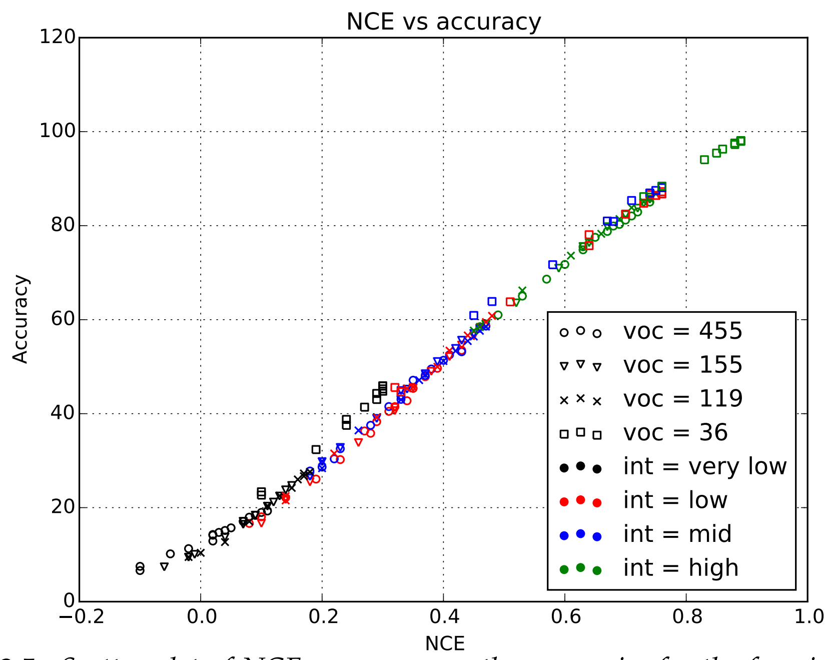

Figure 3.5: *Scatter plot of NCE scores versus the accuracies for the four intelligibility groups and for the four vocabulary sizes. The Pearson correlation coefficient is 0.99.*

is that the optimal vocabulary size depends on the speaker characteristics and on the amount of acoustic data available for adaptation. This is closely related to the "optimal operating point" described in the homeService project (section 2.1.1), suggesting that different speakers will have different optimal operating points. For the Low-Mid intelligibility group, a 36 command vocabulary seems to be a good operating point, as it reaches a good convergence performance after collecting only 300 words. The performance when the ASR is not adapted, however, could be insufficient to maintain the user engaged, which is a key issue in hS systems. POMDP-based dialogue management, due to its increased robustness in ASR challenging environments, could help to improve the system performance in this stage, keeping the user engaged until enough acoustic data has been collected. Nevertheless, 36 commands are enough to control a simple but useful system and it would avoid the need to collect large amounts of enrolment data, which may be infeasible for a dysarthric user. In addition, most of the speakers recruited for the hS project are within the Low and Mid intelligibility groups (Christensen et al., 2015). Because of all the aforementioned reasons, the DM experiments presented in this thesis will be conducted in a 36 command vocabulary simulated hS system. This system is described in section 3.2.

3.1.3 Dialoge Management differences with other Spoken Dialogue Systems

Most of the existing assistive VUIs use rule-based dialogue managers to control the dialogue flow (Christensen et al., 2013b, Vacher et al., 2011, 2015). The main reason for this is that the interaction with these systems is simple enough to hand-craft the set of rules to control them. In the case of VUIs developed for dysarthic speakers, however, a mechanism to deal with the poor ASR performance can be very useful. The POMDP dialogue management framework has been shown to improve the interaction performance in poor ASR conditions (Young et al., 2013), thus it is a very interesting option to control the dialogue flow in the hS environment.

The POMDP-DM framework, however, has mostly been studied in the context of SDSs designed to provide information about a specific topic, such as restaurant information (Young et al., 2010) or bus schedule information (Raux et al., 2005). The analysis done in section 3.1.2 suggests that, when modelling the DM of homeService as a POMDP, some differences with typical SDSs have to be taken into account:

- There are large differences between the speech characteristics of the users. Therefore, a dialogue manager that works well with one user might not work well with another. A framework which is portable for different users is necessary.
- The environment dynamics vary over time. This means that the DM model needs to be adapted to the environment changes as they occur.
- The system will be used by a single speaker over a long period of time. Therefore, larger amounts of user specific dialogue data will be available as the user interacts with the system. This opens the possibility for online user adaptation.

In the following sections, a POMDP model suitable to be used in assistive VUIs is presented, as well as a dialogue framework to test the performance gain that can be obtained using POMDP dialogue management in a homeService system.

3.2 Dialogue design and evaluation framework for hS environment

To evaluate the performance of POMDP-based dialogue managers, a framework in which dialogues with an hS system can be carried out must be developed. In this section, a simulated homeService system with a vocabulary of 36 commands designed to control several devices in a home[1] is presented. In addition, tractable dialogue state representations for these kind of systems are proposed and a set of *simulated dysarthric users* is developed to interact with the system.

3.2.1 Dialogue ontology

As mentioned in section 2.5.1, an *ontology* defines the set of devices the system can control and its functionalities. Figure 3.6 shows the dialogue ontology used by the hS system presented in this section. The users can control four devices (“tv”, “light”, “bluray” and “hi-fi”) by selecting the functionalities they want to control (“on”, “guide”, etc.), which sometimes can take specific values (“up”, “five”, etc.). The possible user goals in this system are defined by the paths to leaf nodes in figure 3.6 (e.g. “tv-channel-two”, “light-on”, “bluray-menu-right”). There is a total of 63 goals. Note that many of the values and functionalities are repeated for several devices (e.g. “on”, “up”), meaning that the system can perform more actions (goals) than the number of commands used to control it. The ASR vocabulary also includes meta-commands such as “yes”, “no” and “back”.

3.2.2 Tractable dialogue state representation for hS

One of the most important tasks when implementing a dialogue manager is the design of the dialogue state representation. This representation is usually derived from the *ontology*, which defines all the actions that the dialogue system can perform (figure 3.6). Designing the dialogue state representation, however, usually requires some sort of hand-crafting (Raux et al., 2005, Young et al., 2010). As a probability distribution over this representation (the *belief state*) is the input for the dialogue policy, it has to be precise enough to distinguish between all the different dialogue situations possible. On the other hand, it also has to be represented in a structure as compact as possible to maintain the system tractability, because in some systems the number of different dialogue situations can be extremely large. In the following subsections two tractable dialogue state representations for hS systems are proposed.

Slot-based architecture

Slot-based architectures are the most common approach taken by SDSs to represent the dialogue state (Henderson et al., 2014a, Williams et al., 2013). This repre-

[1]The architecture of this system is similar to the architecture of the system described in Christensen et al. (2013b).

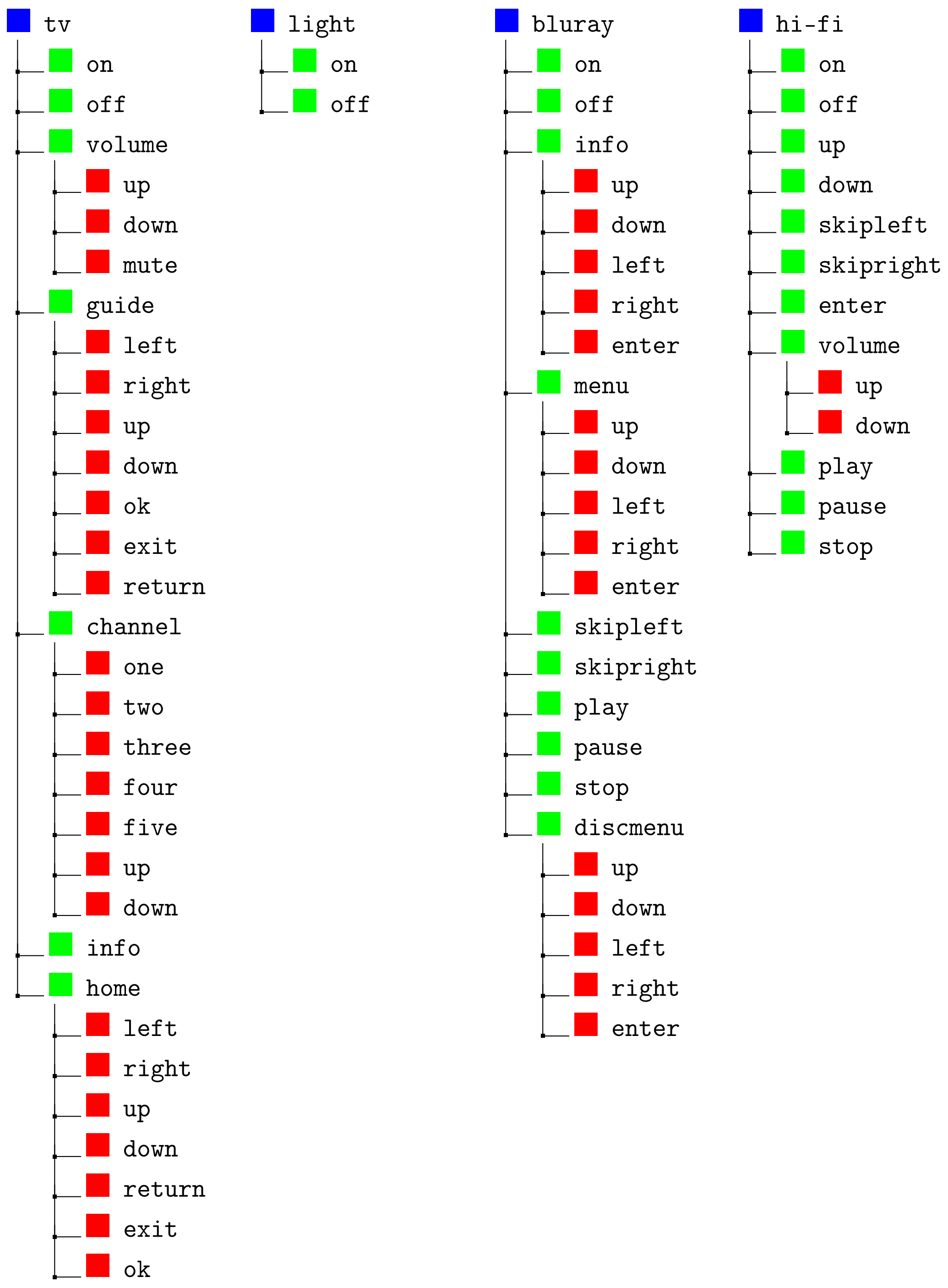

Figure 3.6: *The complete control tree (ontology) of the simulated hS system. The blue squares represent the devices, the green ones the functionalities of the devices and the red ones the values these functionalities can take.*

sentation is inspired by slot filling systems (Mesnil et al., 2013, Price, 1990), where, firstly, a set of slots is defined. Then, a set of mutually exclusive values is defined for each slot and in each turn, each slot will take a value from its corresponding set. In slot-based architectures, the value of each slot is usually inferred independently in each dialogue turn and the *joint* dialogue state is defined by combining the values of the different slots. In other words, the inference of the dialogue state is factorised into the inference of the value for each of the slots, assuming conditional independence between slots. One advantage of using this representation in information gathering dialogue systems is that the slots can be easily derived from the ontology, reducing the amount of hand-crafted effort required in the design of the dialogue state representation.

In the hS environment, however, it is not straight forward to define the slots, because of the tree structure of its ontology. One possible way to do it is to define a slot for each level in the command tree (represented by different colors in figure 3.6) and define the set of values of that slot as all the possible commands in that level. This approach has a couple of shortcomings: when joining the values of each slot, the resulting dialogue state might be invalid (e.g. "lamp-channel-on"). In a state tracker that outputs a probability distribution over values of each slot, this can be fixed by discarding the invalid joint dialogue states and renormalising. Another shortcoming arises from the fact that the same command can occur in two different levels of the tree (e.g. "up"). Contrary to other SLU systems (Henderson et al., 2012, Mairesse et al., 2009), the ASR N-best output does not specify to which slot the commands correspond. In this case, the dialogue state tracker must rely on the dialogue history to infer to which slot each command corresponds.

The slot-based architecture has been shown to be a good state representation approach for several dialogue managers (Raux et al., 2005, Thomson and Young, 2010). However, model-based POMDP solving algorithms are usually intractable for real sized systems when following this approach (Williams and Young, 2007). Therefore, the experiments presented in this chapter (and in chapter 4) will use the tree-based dialogue state representation presented in the next subsection. The slot-based approach is the dialogue state representation used for the dialogue state tracking experiments presented in chapter 5.

Tree architecture

In the rule-based dialogue manager used by the hS systems installed in real users homes (Christensen et al., 2015), the user can navigate through the system in a hierarchical fashion. For instance, if the user wants to change the TV channel to "five", the sequence of words to utter will be *"TV"*, *"channel"*, *"five"*. This mode

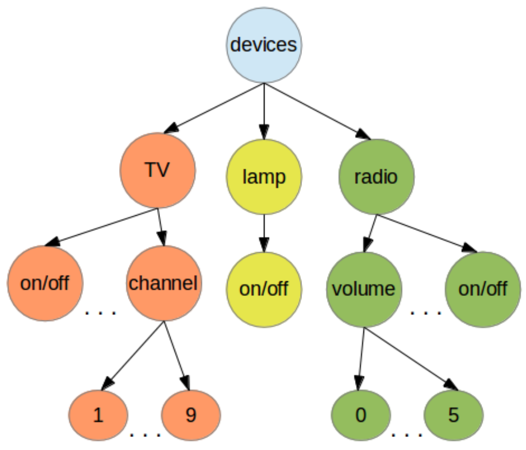

Figure 3.7: *Example of a hierarchical navigation tree used to control a homeService system.*

of interaction is known as *system initiative*, where the system prompts a series of commands the user can utter thus constraining the number of actions the user can take. The contrary approach would be *mixed initiative* systems, where the user can utter the actions in any order, and the system usually intervenes to ask for missing information pieces or clarifications. Usually POMDP dialogue systems are designed to interact in a mixed initiative fashion.

As the most critical issue to deal with in a hS system is the high ASR error rate, it is possible to sacrifice the mixed initiative capability of the system to reduce the complexity, while still using the POMDP framework to increase the robustness against poor ASR performance. To do so, the hierarchical structure of the system is used to organise the dialogue manager: the control architecture of the system is organised in a tree setup as shown in figure 3.7. The system is then modelled as a finite state automaton, where each node represents either a device (e.g. “TV”), a functionality of that device (e.g. “channel”), or actions that trigger some change in one of the devices (e.g. “1”, child of “channel”, will change the TV to the first channel). When the system transitions to one of the terminal nodes, the action associated with this node is performed, and subsequently the system returns to the root node. In the remaining nodes, the user may either say one of the commands available in that node (defined by its children nodes) to transition to them, or say the meta-command “back” to return to its parent node. Therefore, each non-terminal node can be considered as an independent sub-dialogue, whose task is to decide to which node the system will transition to. This idea is similar to hierarchical RL (Cuayáhuitl et al., 2010, Dietterich, 1998), with the difference that only the decisions taken in the nodes of the hierarchical tree are optimised by RL, while the navigation through the tree follows simple rules. The idea behind this architecture is that the "complex" decisions[1] taken by hS systems, are related to accept the last command observed from the user as true or ask for clarifications, while the decisions related to the navigation through the system tree are simple once the true user command is known. Following this architecture the interaction mode will be *system initiative*, because the number of actions the user can take is constrained by the commands shown by the system in the tablet.

As mentioned before, the main shortcoming of this approach is that the system cannot interact in a mixed initiative fashion. However, it considerably reduces the complexity of the model, as the state space size for each sub-POMDP will be the number of children of the node plus one[2] (12 in the worst case). This makes model-based POMDP algorithms tractable. This approach is used for the experiments done in this chapter and in chapter 4.

3.2.3 Simulated homeService environment

Deployment and evaluation of spoken dialogue systems is often very costly in terms of time and resources. When the users suffer from dysarthria and disabilities, the evaluation is even more costly because for these users, speaking requires a greater effort and they can easily get tired. The development of SDSs requires many iterations of testing, because hyperparameters of these models need to be tuned and different DM models need to be compared. This is not a problem in supervised learning tasks, such as ASR or SLU, because any number of offline tests can be performed in a held-out evaluation set to compare different models and settings. However, DM is an *optimal control of stochastic dynamic systems* task (Bertsekas, 1995). In such a control task, the actions taken by the user depend on the actions taken by the system and vice-versa. This means that a dialogue corpus obtained using a DM model cannot be used to test a different DM model, thus any new model needs to be evaluated by directly interacting with the environment – i.e. having dialogues with the user. This is obviously very costly for SDSs[3] and it is even more costly if the users have disabilities. An approach often taken in control tasks for the early deployment of systems and parameter tuning is to create a *simulated environment*, a model that can simulate the signals that would be observed from the environment by the agent (Asada et al., 1999, Matarić, 1997). In the case of SDSs, this environment is often called a Simulated User (SU) (Eckert

[1]"complex" not only because the dysarthric speech, also because these decisions should be different depending on the dysarthria severity of the user.

[2]The extra dialogue state is the state "back", which denotes the users intention of transitioning to the parent node.

[3]Even if, thanks to crowdsourcing tools such as the *Amazon mechanical turk*, the cost of evaluating SDSs has been reduced (Jurčíček et al., 2011).

et al., 1997, Georgila et al., 2006, Schatzmann et al., 2007a, Thomson et al., 2012).

The objective of this chapter is to evaluate to what extent can the POMDP-DM framework improve the performance of a personalised VUI, as well as making this framework usable in the scenarios described in section 3.1.3. To do so, the most appropriate[1] approach is to develop a simulated environment that can emulate the dynamics that will be found in the hS scenarios. This environment has to be able to simulate the interaction with dysarthric speakers with different severities as well as the interaction with an ASR system whose performance improves over time (like in figure 3.3). This is equivalent to developing a set of different simulated environments (corresponding to different *speaker-ASR pairs*), in contrast to the usual SDS approach in which a single environment is designed[2]. Once the simulated environments have been developed, several POMDP-DM models can be compared with rule-based dialogue managers and evaluated interacting in the different environments.

Therefore, a set of speaker specific *simulated users* (SUs) have been built with the purpose of testing the system, where each SU simulates a *speaker-ASR* pair. 15 different dysarthric speakers are simulated (corresponding to the 15 dysarthric speakers in the UASpeech database) and each speaker interacts with 11 different ASRs (corresponding to acoustic models adapted with a number of words that ranges from 0 to 500, with a 50 word increase in each step). This gives a total of 165 *speaker-ASR* pairs (environments) that can be evaluated.

The simulated environment (shown in figure 3.8) is factored into two models: The *behaviour model* and the *ASR channel model*. The behaviour model simulates the decisions taken by the user in each dialogue turn, while the ASR channel model simulates the distortion introduced to the true user action by the ASR channel (which depends on the dysarthria severity and the amount of data used to adapt the ASR). More formally, in each dialogue the user has a fixed goal $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of possible goals. In each turn t, an observation ω_t is generated by the simulated environment given the user goal and the user dialogue history $\mathbf{h}_t^u$ as:

$$P(\omega_t|g, \mathbf{h}_t^u) = P(u_t|g, \mathbf{h}_t^u)P(\omega_t|u_t) \tag{3.1}$$

Where u_t is the true user action or command, $P(u_t|g, \mathbf{h}_t^u)$ is the behaviour model[3], $P(\omega_t|u_t)$ is the ASR channel model and $\omega_t = (\tilde{\mathbf{u}}_\mathbf{t}, \mathbf{c}_t) = (\tilde{u}_{t,1}, ..., \tilde{u}_{t,n}, c_{t,1}, ..., c_{t,n})$

[1]Appropriate because, if every developed model was evaluated with real users, the time spent doing the evaluations would have been infeasible.

[2]Although it is usual to simulate these environments in different ASR noise conditions (Thomson et al., 2012).

[3]Note that as the behaviour model is deterministic, $P(u_t|g, \mathbf{h}_t^u)$ will be 1 for one user action and 0 for the rest, thus there is not need to sum over all user actions.

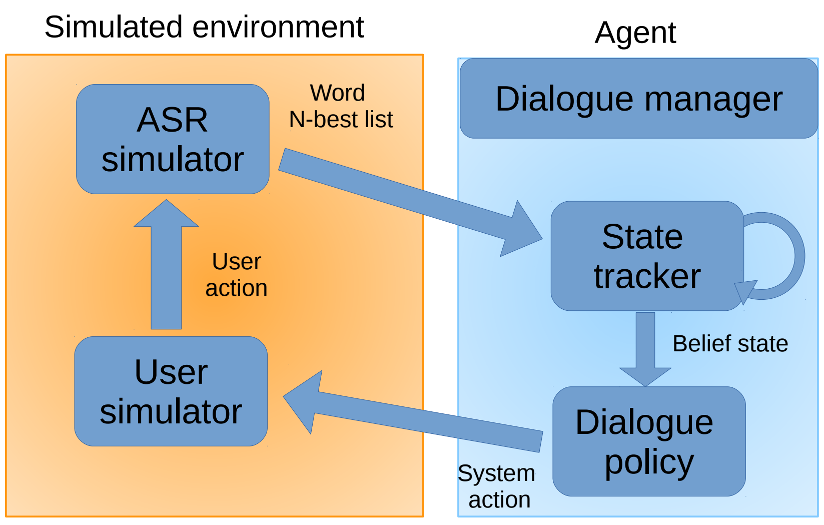

Figure 3.8: *Schematic representation of the homeService simulated environment interacting with the dialogue manager.*

is an N-best list of command hypotheses with normalised confidence scores of length n. The user history $\mathbf{h}_t^u$ is the sequence of actions taken by the user and the actions observed from the system up to turn t.

In the next section the user behaviour model and the ASR channel model are explained in more detail. The set of simulated users described in this section is used in section 3.4 to evaluate several POMDP models and to compare them with rule-based dialogue managers.

User behaviour model

The user behaviour model, $P(u_t|g, \mathbf{h}_t^u)$, decides which is the next action u the user will take given the user goal g and the user dialogue history $\mathbf{h}_t^u$. The user dialogue history is all the information seen in the dialogue so far from the user's point of view, e.g. all the machine actions and true user actions. An agenda-based simulation approach (Schatzmann et al., 2007a) has been followed to build this model, with some simplifications. Firstly, it is assumed that the user will not change his goal during a dialogue: at the beginning of each dialogue, a user goal g is randomly generated (e.g. *TV-Channel-Five*) from the set of possible goals $\mathcal{G}$, and this goal will stay fixed until the dialogue ends. Secondly, the user action is assumed to depend only on the last system action rather than on all the dialogue history. Following these assumptions, the behaviour model is approximated as:

$$P(u_t|g, \mathbf{h}_t^u) \simeq P(u_t|g, a_{t-1}) \tag{3.2}$$

where a_{t-1} is the action taken by the system in the previous turn.

The behaviour model is deterministic and the generation of the user action follows the following set of rules:

- If the system is waiting for a command input, the SU will say one command of the goal (not previously said) per turn.
- If the system asks the user to repeat his last command, the user will do so.
- If the system has transitioned to a wrong state, the user will say the meta-command *"back"*[1].
- If the system asks a confirmation question, the user will answer yes/no.

It is assumed that the user knows how the control system works, (i.e. knows which command to utter in each turn to fulfil his goal) and is fully collaborative with the system (e.g. if the system asks to repeat his last command, the user will repeat it, even if it is asked three times to do so). In the following experiments, the behaviour model of each user is assumed to be the same[2]. The reason for this is that the research question trying to be answered depends on the user dysarthria severity and on the amount of data used to adapt the ASR, not in the user behaviour. In appendix B, the configuration of the user behaviour model is explained in more detail. The dysarthria and ASR simulation models (the ASR channel) are explained in the next subsection.

ASR channel simulation model

The true action or intention of the user has to travel through a channel before reaching the dialogue manager as an observation. This channel is composed by the user's speech generation system (which depends on the dysarthria severity) and the ASR (which depends on the amount of data used to adapt it). The ASR channel model, $P(\omega_t|u_t)$, simulates the noise introduced to the true user action or user command u_t by this channel. In other words, the ASR channel model converts

[1]This can only happen if the tree-based dialogue state representation is used.

[2]In the state tracking experiments performed in section 5, different user expertise levels are simulated by including a probability for the user to say a wrong command, simulating that the user does not totally know which commands have to be used to control the system. Refer to appendix B for a more detailed description.

the true user action u_t into an N-best list of noisy user action recognitions with confidence scores ω_t.

A different ASR channel simulation model is trained for each *speaker-ASR* pair and its parameters are learnt from the statistics obtained from the speech recognition experiments done in section 3.1.2. The ASR channel model is trained using half of the N-best outputs obtained in these experiments[1]. Due to the small amount of data available for training, directly modelling the distribution of the N-best lists with confidence scores would be infeasible. To deal with the data sparsity problem, the ASR channel model is factorised by assuming conditional independence between the commands of the N-best list and the confidence scores given the true user action:

$$P(\omega|u) = P(\tilde{\mathbf{u}}, \mathbf{c}|u) \simeq \prod_{i=1}^{n} P(\tilde{u}_i|u) \prod_{i=1}^{n} P(c_i|i) \tag{3.3}$$

where i is the position in the *N-best* list. Following the approach taken by Williams et al. (2005), two probability distributions for the confidence scores are learnt: $P(c_i|i) = P_{cor}(c_i|i)$ if $\tilde{u}_i = u$ (if the confidence score corresponds to the true user action) and $P(c_i|i) = P_{inc}(c_i|i)$ otherwise. The distribution for the generation of words in the N-best list $P(\tilde{u}_i|u)$ is learnt from the weighted confusion matrix statistics obtained from the experiments in section 3.1.2. However, due to the factorisation, the model might generate *invalid* N-best lists (e.g. repeating a command in two positions of the list). Therefore, after generating ω_t, the N-best list must be post-processed to satisfy three constrains:

- Two commands cannot be repeated in the N-best list: $\tilde{u}_i \neq \tilde{u}_j \, \forall \, j \neq i$
- Each confidence score must be less or equal than the confidence scores above it in the N-best list: $c_i \geq c_j \, \forall \, i > j$
- The confidence scores must sum to one: $\sum_{i=1}^{n} c_i \leq 1$

A more detailed description of the ASR channel simulation model is given in appendix B.

Note that the generation of ω_t is independent of the time step t; the ASR or the user do not produce different outcomes depending on the dialogue turn. This is the case in many ASR systems, although some systems are beginning to include context dependant ASR systems; systems where the ASR decoding depends on the dialogue state (Jonson, 2006).

[1]The other half is used to train the POMDP models used in section 3.3. This is done to maintain independence between the training and testing data

Sampled ASR simulation

An alternative approach to sampling ω_t given the true user action by training a generative model, is to directly sample from the actual N-best list produced by the ASR in the experiments of section 3.1.2. This method will generate more realistic ASR simulations since it does not have to assume independence between the N-best list and the confidence scores. However, the amount of data available in the UASpeech database is small, so the variability of the sampled N-best lists will be too small (the same observations would be sampled again and again). To solve this, a data augmentation approach can be taken by increasing and decreasing the speech rate of the recordings by a small amount, generating more (synthetic) data. This approach has been shown to improve ASR accuracy when used to increase the amount of training data in low resourced languages (Ragni et al., 2014). This approach is used in the dialogue state tracking experiments performed in chapter 5, because some of the experiments also need to sample the phone posterior probabilities outputted by a neural network feature extractor.

3.3 POMDP-based dialogue management for hS

The main concern when interacting with dysarthric speakers is the low performance of the ASR, which is not only worse than the average, but also has a high variation between different speakers. The POMDP-DM framework has been shown to improve dialogue interaction robustness when dealing with high error rate ASR systems (Young et al., 2013), so it naturally fits into dialogue interaction with dysarthric speakers. It is also a data driven method, meaning that the models can be learnt from (or adapted with) user specific data, which can deal with the problem of high variability between speakers. In this section, a tractable model-based RL method to optimise the policy of a dialogue manager in a command based VUI is proposed.

Following the tree-based dialogue state representation proposed in section 3.2.2, the scalability (and data sparsity) problems of model-based RL methods can be solved. To do so, an independent POMDP is defined for each non-terminal node in the control tree with the following specifications:

- The set of states $\mathcal{S}$ is the set of available commands (children) in the node, plus a state for the meta-command "back", which denotes that intention of the user is to transition to the parent node.
- The set of actions $\mathcal{A}$ is composed by the actions to transition to each child node, a "back" action (which performs a transition to the parent node) and

an “ask” action which requests the user to repeat his last command.

- The set of observations Ω is the set of commands the ASR is able to recognise (the command vocabulary).

The main advantage of this DM design, is that it enables the study of the robustness provided by the POMDP-DM framework against ASR errors, while maintaining the system tractability. Following this system design, the flexibility given by mixed initiative systems is sacrificed in order to make the model computationally tractable and to require less data to train the model. Recalling section 2.4.1, POMDPs are factorized into an observation model $O(\omega', s', a)$ and a transition model $T(s, a, s')$. As it is assumed that the user will not change his goal during a single dialogue and will collaborate with the system, the transition model is trivial and can be hand-crafted (Williams and Young, 2007). Therefore, only the observation model needs to be trained from data. In the next subsections the dialogue state tracking model and the policy optimization algorithms are explained.

3.3.1 Generative Dialogue State Tracking

If it is assumed that the user does not change his or her goal[1], the transition function in equation 2.5 can be modelled as an identity function: $P(s_t|s_{t-1}, a_{t-1})$ is equal to 1 if $s_t = s_{t-1}$ and 0 otherwise. Then, assuming that the observation ω_t is independent of the previous system action and that the commands in the N-best list are independent between them, $P(\omega_t|s_t, a_{t-1})$ can be approximated as the sum of the probability of observing the hypothesis $\tilde{u}_t$ when the actual user command[2] is s, weighted by the confidence score of that command in the observed N-best list $c(\tilde{u}_t)$. After these assumptions, the typical belief update equation (equation 2.5) can now be rewritten as:

$$b(s_t) = k \cdot b(s_{t-1}) \sum_{\tilde{u}_t} P(\tilde{u}_t|s_t)c(\tilde{u}_t) \tag{3.4}$$

This approach deals with the tractability problems that many model-based algorithms face in dialogue management, where the number of observations can be infinite (Gašić, 2011, Williams et al., 2005).

[1]this is a reasonable assumption in some systems, e.g. (Williams and Young, 2007)

[2]In this case, the user command corresponds to the true user intention in the dialogue, which is in turn the dialogue state.

3.3.2 Model-based Reinforcement Learning policy optimization

If the transition model, the observation model and the reward function have been defined, the optimal policy can be found by using point-based value iteration RL algorithms (Pineau et al., 2003, Smith and Simmons, 2012) (see section 2.6.1). The main shortcoming of these methods is that their complexity grows exponentially with the number of states of the POMDP, which makes them usually intractable for DM tasks. Due to the tree-based dialogue state representation, however, the number of states of each sub-POMDP is small enough to be solved by value iteration methods.

3.4 DM experiments in a simulated hS environment

To evaluate the improvement that can be obtained from using the probabilistic dialogue management framework presented in section 3.3 in a VUI personalised to dysarthric speakers, a set of experiments are carried in the simulated hS environment presented in section 3.2.3. A rule-based DM approach, similar to the one being used by the hS systems installed in the users homes, is developed to serve as baseline. To evaluate the performance of the POMDP framework in different environments, the systems are tested with the 15 different simulated dysarthric speakers interacting with the ASR adapted with increasing amounts of data. POMDP dialogue managers trained with different data configurations (e.g. speaker specific data only, data from other speakers, data from different ASRs, etc.) are also compared. These different models are trained to get an insight of the performance of the POMDP-DM framework in scenarios that are specific to VUIs personalised to dysarthric speakers.

3.4.1 Experimental setup

The system uses the tree-based dialogue state representation described in section 3.2.2. For the POMDP-based dialogue manager, the state $\mathcal{S}$, action $\mathcal{A}$ and observation Ω set for each non-terminal node is defined as explained in section 3.3. The immediate reward function in each node is defined as follows: -1 for the "ask" action, +10 for any transitioning action if it corresponds to the user goal (correct terminal action for the node) and -10 otherwise. This is a reward function modelling approach widely used in task oriented dialogue management (Gašić, 2011, Pietquin et al., 2011, Williams and Young, 2007, Young, 2000) which aims to trade-off between dialogue success rate and dialogue length. The ratio between the reward for successful dialogue completion and the turn penalty determines the

Table 3.2: *Example homeService dialogue using the tree-based POMDP architecture. The goal of the dialogue is "TV-Volume-Up".*

Dialogue starts in node "Devices"

Sub-dialogue "Devices"

User	*TV* (Speaks the command "TV")
System	*Node-TV* (Dialogue transitions to node "TV")

Sub-dialogue "TV"

User	*Channel* (Speaks the command "Volume")
System	*Ask* (Requests the user to repeat his last command)
User	*TV* (Repeats his last command)
System	*Node-Channel* (Dialogue transitions to node "TV-Volume")

Sub-dialogue "Volume"

User	*One* (Speaks the command "Up")
System	*Ask* (Requests the user to repeat his last command)
User	*TV* (Repeats his last command)
System	*Node-One* (Performs action *TV-Volume-up*)

As an action has been taken in a terminal node, the dialogue ends. As the action taken in the terminal node matches the goal, it is a successful dialogue.

importance given to each of these dialogue factors. To find the optimal POMDP policy, the point-based value iteration algorithm is used (Pineau et al., 2003), implemented using the *zmdp* toolkit[1]. Table 3.2 shows an example homeService dialogue using the tree-based architecture. In appendix A more example homeService dialogues can be found.

Rule-Based dialogue manager

The objective of the following experiments, is to compare the performance of the POMDP-based dialogue managers to the rule-based ones – i.e. the dialogue managers used in the hS systems currently installed in users homes. To do so, a rule-based DM which follows the approach taken by these hS systems is used as baseline in the following experiments. In each turn, this dialogue manager uses only information from the local confidence scores to control the dialogue flow. It performs the top command hypothesis of the observed N-best list if its confidence score is higher than a threshold, or it asks to repeat the last command otherwise. This threshold is optimized for each *speaker-ASR* pair using grid-optimization

[1]`https://github.com/trey0/zmdp`

methods[1]. If the top hypothesis in the N-best list is not a valid command in the current node, the system will ask the user to repeat the last command.

POMDP dialogue manager variations

As explained in section 3.3, only the parameters of the observation function of each sub-POMDP need to be learnt from data. In the following experiments, these parameters are learnt from the statistics obtained in the speech recognition experiments presented in section 3.1.2. More precisely, $O(\omega', s', a) = P(\omega'|s', a)$ is assumed to be independent of a and the time-step, becoming $P(\omega|s)$. This is a reasonable assumption, because the ASR does not depend on the system action when it recognises the user command. Then, for each $\omega \in \Omega$ and $s \in \mathcal{S}$, $P(\omega|s)$ is defined as the normalised sum of confidence scores obtained for the observation ω when the dialogue state (the true user intention or user goal) was s. In other words, $P(\omega|s)$ is defined as the "weighted confusion matrix" whose statistics are obtained from the experiments presented in section 3.1.2.

In an optimal scenario for personalised DM, these statistics would be learnt from the speech recognition results of the speaker which will use the system. To evaluate this scenario, a model trained with data of the speaker which will use the system, named *Speaker Dependant* (*SD*), is tested in the experiments presented in the next section. However, the scenario where there is enough data from the user to train the systems is rare, specially in a newly set-up system. A scenario where there is only data available from other speakers will be more common. To evaluate this case, a model trained with data from the other 14 speakers, named *Speaker Independent* (*SI*), is also tested. Take into account that the amount of data used to train the *SI* model is on average 14 times larger than for *SD*, because data from 14 speakers is be used instead of from only one[2].

In order to analyse more complex scenarios that can arise in a personalised VUI, another two training data configurations are also evaluated. To investigate the effect of the variations in the environment (e.g. ASR improvement over time) a model is trained with speaker dependent data collected with an non-adapted ASR. Then, to evaluate the effect of the increasing difference between the training and test environments, this model is tested interacting with the 11 ASRs adapted with increasing amounts of data. In the following experiments, this model is named *SD no adaptation*. Finally, another scenario likely to happen in hS is also

[1]This is because the confidence score distributions are very different for each speaker-ASR pair, depending on their dysarthria severity.

[2]The N-best recognition results for half of the recordings in the UASpeech database are used per speaker, as the other half is used to train the simulated users. Check the statistics in table 3.1 to see the data amount per speaker.

evaluated: The DM model is initially trained with speaker independent data (as in *SI*), and in every increasing amount of ASR data environment, it is trained with the same increasing amount of speaker dependent data (plus the speaker independent data). This model, named *SI+SD adaptation*, represents an online adaptation scenario: a scenario where the DM models are adapted as data from the user becomes available.

Tree-based dialogue manager evaluation

Due to the variable depth tree structure of the spoken dialogue system, the sum of the accumulated rewards obtained in each sub-dialogue is not a good measure of the overall system performance. For example, a successful dialogue whose goal is "TV-Channel-On" would get an accumulated reward of 30 while a dialogue whose goal is "Light-On" only 20. Using the average accumulated reward in each dialogue is not a good measure neither, because if the dialogue gets stuck in a loop going back and forth between two sub-dialogues, the extra amount of turns spent in this loop would not be reflected in the average of rewards. To evaluate the dialogue performance in the following experiments, a full dialogue based approach is taken: the reward obtained is defined as 20 if the dialogue is successful (or 0 otherwise), minus the number of turns spent to finish the dialogue. To avoid infinite dialogues, if a dialogue takes more than 30 turns, the dialogue will stop and it will be considered unsuccessful.

3.4.2 Experimental results

To compare the performance of the different dialogue managers presented in section 3.4, a set of experiments has been carried out in the simulated hS environment. Each speaker-ASR pair is evaluated simulating 5000 dialogues[1] with each dialogue manager (including the rule-based baseline) and two metrics are measured: the dialogue success rate, whereby a dialogue is considered successful if the terminal action carried out by the system matches the user goal[2], and the tree-based dialogue reward defined in the previous section.

Figure presents 3.9 the absolute results in success and reward (with variance), for four dysarthric speakers corresponding to speakers from different intelligibility groups (*F03*, *very low*, black; *F02*, *low*, red; *F04*, *mid*, blue; and *M08*, *high*, green).

[1]Due to the large number of trials conducted with each dialogue manager, the standard error of the mean is various orders of magnitude smaller than the difference between the results and therefore it is not plotted. This is also the case for the experiments in chapters 4 and 5. In some plots, however, the standard deviation is plotted as it is indicative of the variability between speakers.

[2]In the tree-based architecture, this means transitioning to the terminal node corresponding to the user goal.

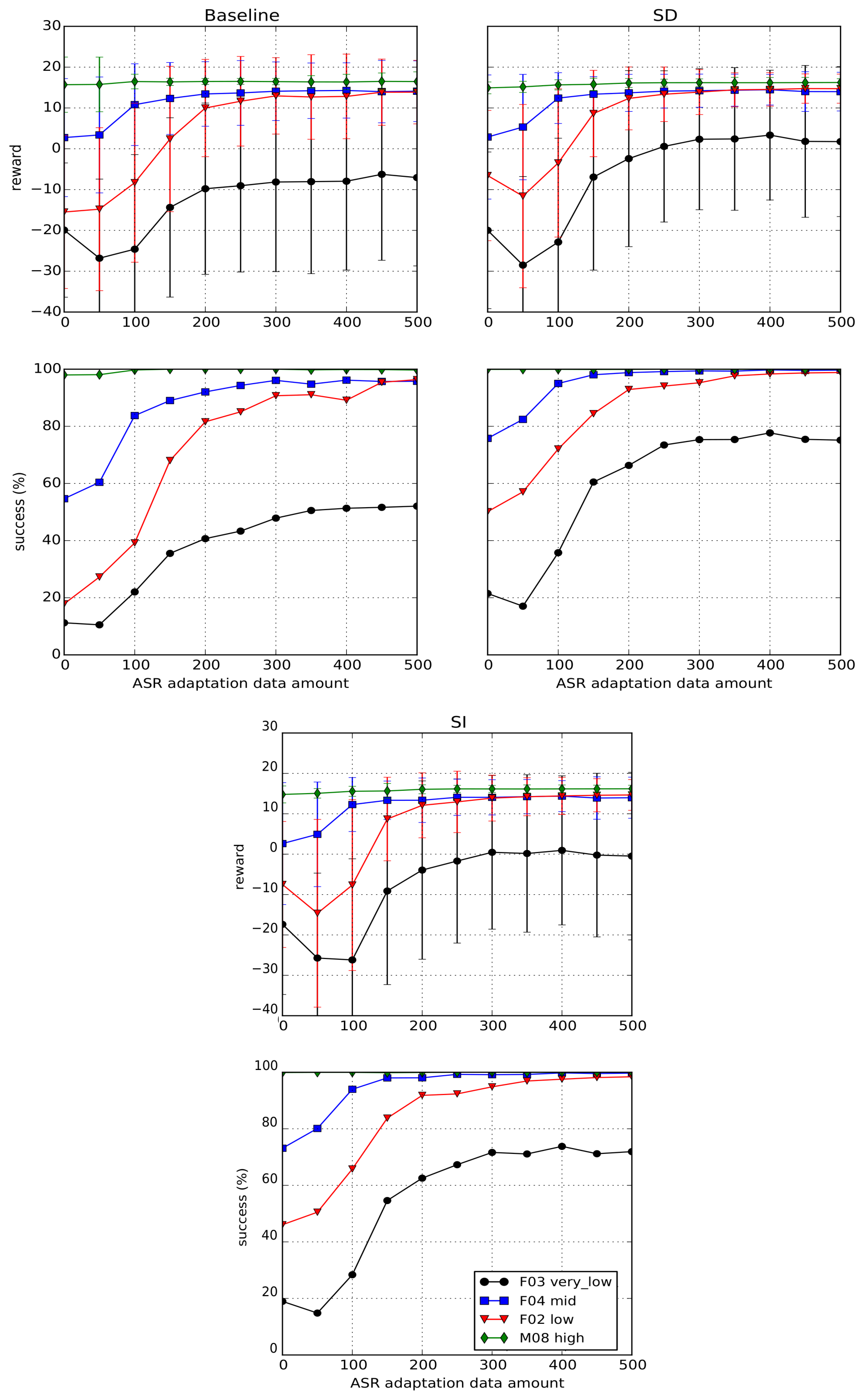

Figure 3.9: *Reward and success rate for the rule based dialogue manager (baseline), speaker dependent manager (top right) and speaker independent manager (bottom) for four speakers. The error bars represent the standard deviation of the reward. The x axis represents the amount of word recordings used to adapt the acoustic models.*

The x axis represents the number of single-word recordings used to adapt the ASR. If the results for the different speakers are compared, it can be seen that the performance differs for each intelligibility group. Firstly, observe how *M08* shows an almost perfect performance for any amount of ASR adaptation data. For the other three speakers, in contrast, it can be seen how the improvement of the dialogue performance is very correlated with the improvement of the ASR accuracy shown in figure 3.3, converging after 300 words. After convergence, the performance for *F02* and *F04* becomes very similar to the performance of *M08*. *F03*, however, shows a performance much lower than the other speakers, with a very high variance in the reward. In the case of the low and mid speakers, the high variance behaviour is also observed when the ASR is not adapted, but the variance decreases as the amount of ASR adaptation data increases.

If the different dialogue managers are compared, it can be seen how the POMDP-based ones clearly outperform the baseline for the Very Low, Low and Mid speakers. The *SD* manager also outperforms the *SI* one, but only by a small margin. The performance of all the dialogue managers improves as the ASR adaptation data amount increases, thus the absolute results do not show a clear comparison of the different managers. The main interest of these experiments is to evaluate the improvements obtained by each POMDP-based dialogue manager compared to the baseline in each ASR environment, which can be seen more clearly if the relative improvement with respect to the baseline is plotted.

Figure 3.10 shows the relative improvement with respect to the baseline of the *SD, SI, SD no adaptation* and *SI+SD adaptation* dialogue management models (shown in different colors). In order to evaluate the performance of all the speakers, each column shows the mean relative reward and success performance of the Very Low, Low+Mid and High intelligibility speaker groups respectively. In this figure, the comparison of the different dialogue managers for the different intelligibility groups interacting with ASRs adapted with different amounts data can be done more clearly. In the top-left plot (the very low intelligibility speakers) two different regimes can be observed depending on the amount of ASR adaptation data. In the first regime (less than 300 adaptation words), the relative performance is very noisy (specially in the reward), varying abruptly for different ASR adaptation amounts and for different models. In the second regime (more than 300 adaptation words) the results are much more stable, showing a high improvement for all the models except for *SI+SD adaptation*. For the high intelligibility speakers (the bottom plot), in contrast, the relative performance gain of POMDP-based models with respect to the baseline is almost none.

The most interesting plot, however, is the one corresponding to the low and

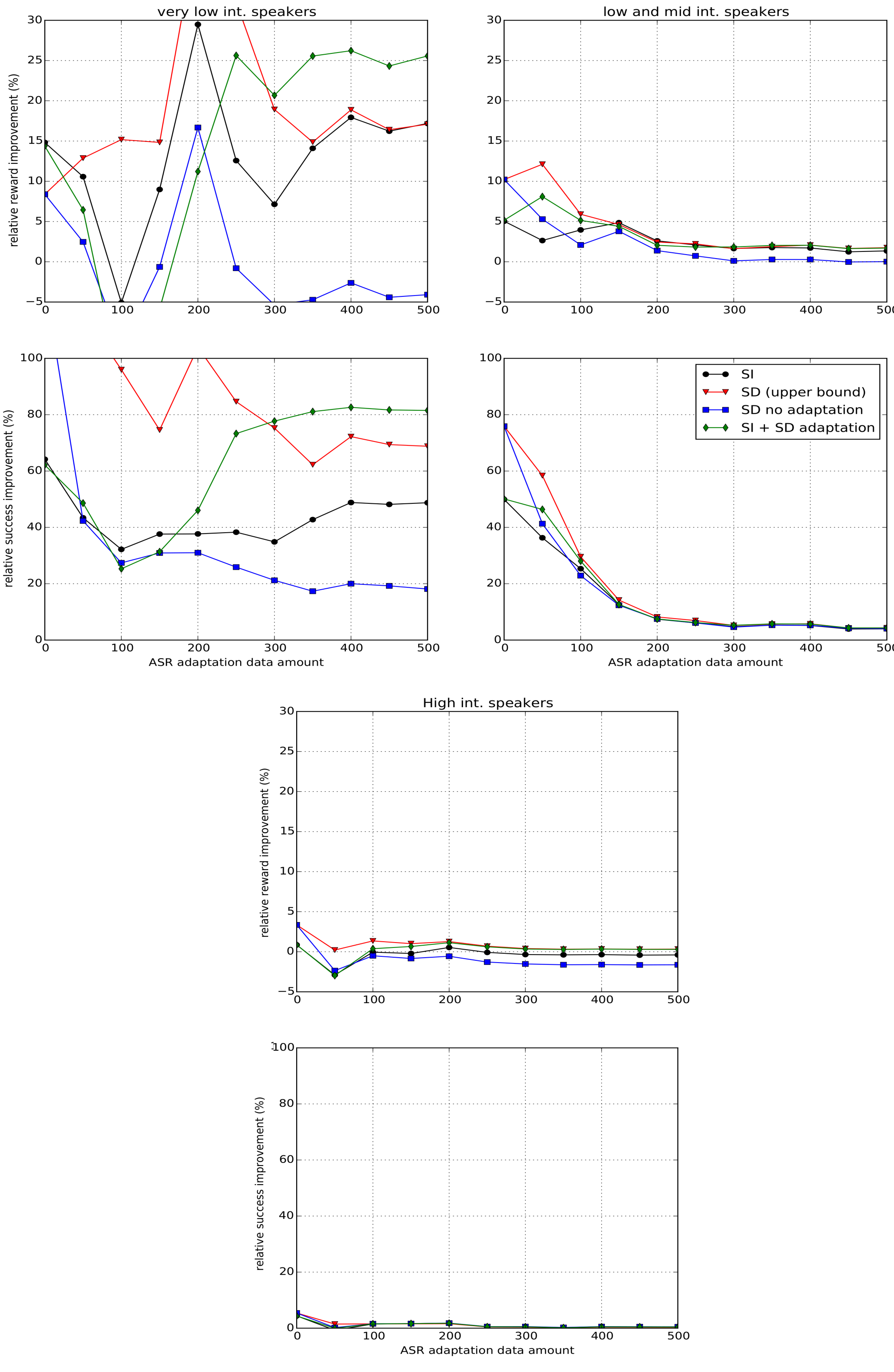

Figure 3.10: *Mean relative performance respect to the baseline for reward and success rate for the different intelligibility groups and for the four DM variations evaluated. The scale of the y axis is kept the same for a clearer comparison.*

medium intelligibility speakers (top right), because most of the users recruited for the hS project are within this intelligibility range (Christensen et al., 2015). The performance of all the POMDP-based dialogue managers when the amount of ASR adaptation data is small is much better than the performance of the baseline. As it was expected, the *SD* model shows the best performance, but the performance of the *SI* model still outperforms the baseline by a large amount. In the case of the *SI+SD adaptation* model, which is a more realistic scenario than the *SD* model, its performance is as good as the *SD* model after collecting only 100 words from the user. Finally, observe that the *SD no adaptation* model behaves similarly to the other dialogue managers when it interacts with a non-adapted ASR, but its performance quickly falls below the baseline as the ASR is adapted with increasing amounts of data.

Experiment conclusions

In figure 3.9 it could be seen that the performance of POMDP-based dialogue managers is very dependent on the speakers' dysarthria severity. On the one hand, the high intelligibility speaker, *M08*, showed an almost perfect performance for any amount of ASR adaptation data. This is because in a 36 word vocabulary, the ASR accuracy for high intelligibility speakers is high enough to make almost no mistakes, therefore both rule-based and POMDP-based dialogue managers show an almost perfect performance. The very low intelligibility speaker (*F03*), on the other hand, showed a performance much lower than the other speakers, with a very high variance in the reward. The high variability probably means that, for different user goals, the average length and success of the dialogues is very different. If the ASR yields very low accuracy for certain commands, which can often happen for very low intelligibility speakers, the system might get stuck in some node of the tree-based structure. In a mixed initiative system (e.g. with a slot-based dialogue representation), the value for the other slots could help to better track the problematic commands. In the case of the low and mid speakers, the high variance behaviour observed when the ASR is not adapted is probably has the same underlying reason.

Figure 3.10 showed a clearer comparison of the POMDP-based models with the baseline. The very low intelligibility speakers showed a very noisy behaviour in the regime of small ASR adaptation data amount, but the results became more stable when the amount of adaptation data became larger. Encouragingly, a large relative improvement could be observed in both regimes, but, observing the absolute results plotted in figure 3.9, it can be seen that the absolute performance of the POMDP-based DM is still very poor, especially in the first regime. This performance, however, is still better than the performance of the baseline, which reflects in a very high relative improvement (because the absolute performance of the baseline is extremely low). It is still encouraging to see that the performance increase is higher and stable for larger amounts of ASR adaptation data, but it is not clear if the absolute performance is high enough to make the system useful.

Testing the POMDP-based models with the high intelligibility speakers, on the other hand, showed almost no performance gain with respect to the baseline. As explained before, this is due to the fact that the ASR performance is high enough to have an almost perfect recognition accuracy, so the performance of the rule-based dialogue manager is almost perfect. The performance improvement that can be obtained from POMDP-based models is therefore very small.

The most encouraging result is observed for low and mid intelligibility speakers, for which POMDP-based dialogue managers have a much better relative performance when the amount of ASR adaptation data is small. This corresponds to the initial stages of usage of a system, where it is more important to keep the user engaged to continue collecting acoustic data (recall the "virtuous circle" in section 2.1.1). The improvements can be observed both in success and reward, but they are much higher in success (between 50% and 80% relative improvement). Therefore, when the amount of ASR adaptation data is small, the POMDP framework is increasing the dialogue success rate of the system at the cost of longer dialogues. Then, as the ASR performance increases, the POMDP framework adapts the policy to a more straight forward one.

It is also interesting to see how the performance of the *SI* model still outperforms the baseline by a large amount. This suggests that data collected with different speakers can still be used to train a model for a specific speaker. In the case of the *SI+SD adaptation* model, its performance is as good as the *SD* model after collecting only 100 words from the user, showing that, initialising the DM models with data collected from other speakers and adapting these models online as speaker specific data is collected through interaction, is a very promising approach.

Finally, it could be observed how the performance of *SD no adaptation* is similar to the other DMs when it interacts with an non-adapted ASR, but greatly decreases when the ASR is adapted with increasing amounts of data. This is because of the mismatch of the data used to train the dialogue models (which comes from an ASR with a higher error rate) and the data produced by the simulated environment. This suggests that mechanisms to deal with the variations in the environment must be used.

3.5 Conclusions

The main objective of this chapter, was to evaluate performance gain that can be obtained applying the POMDP-DM framework to VUIs personalised to dysarthric speakers and to identify the modifications needed to apply this framework to this kind of systems. In the experiments presented in section 3.4.2, it was shown that the robustness against poor ASR performance of POMDP-based dialogue managers can greatly improve the performance of these systems, especially in the case of speakers with low to medium intelligibility. For these speakers, the highest improvement was observed when the amount of ASR adaptation data is small, which correspond to the initial stages of usage of the system, when the ASR performance is worse. As was stated in section 2.1.1, this is a critical stage of usage of hS systems, because poor system performance can make the user lose interest in the system and stop using it. At the cost of longer dialogues, POMDP-based dialogue managers can make the user keep the interest in the system when the ASR performance is worse. Then, when enough acoustic data to get a good ASR performance has been collected, the policy can be adapted online to a more straight forward one. This can widen the range of the "optimal operating point" for hS systems explained in section 2.1.1, as the system can perform better over a wider range of ASR accuracy, reducing the amount of times the researchers need to extend the range of devices the system can control.

Another objective of this chapter was to identify and analyse the issues that may arise from using the POMDP framework in systems different to the typically studied information gathering SDSs. Two main conclusions can be drawn from the experiments in section 3.4.2:

- Using data gathered from other speakers to train the models gives a good performance. This is interesting because this situation might be encountered several times in hS like scenarios (e.g. a new system is being set up for a new user and the researchers have data collected from other systems in other users houses). It was also shown that, initialising a DM model with data from other users, and then adapting this model online to the specific user with data gathered through interaction, reduces the amount of speaker specific data needed to get a "speaker specific" performance. This suggests that more sophisticated online adaptation techniques could improve the system performance further.

- As the environment changes, the improvements obtained from using POMDP-based DM decrease. However, this improvement reduction is gradual, suggesting that data collected in "similar" environments can still be useful. Due to the cost of collecting data with dysarthric speakers, it would not be feasible to collect large amounts of dialogue data each time the environment changes, so methods to effectively use the data coming from different environments should be studied.

To make model-based RL approaches tractable, a tree-based hierarchical dialogue state representation and POMDP architecture was developed. However, several assumptions had to be made and the dialogue manager lost the ability to interact in a mixed initiative fashion. The assumptions made the dialogue manager a highly structured model, with only a few parameters to learn. It has been shown that these assumptions might hold in small scale systems (Williams et al., 2005), but as the system size increases, more complex structures need to be hand-crafted (Thomson and Young, 2010). Model-free RL approaches (Gašić, 2011) could be a better option for larger systems, because they would rely on less assumptions to model the dialogue policy. Slot-based architectures (Henderson, 2015a) could also let the system interact in a mixed initiative fashion increasing the naturalness of the dialogue.

The need for a fast evaluation framework was also analysed in this chapter. Because of the cost of evaluating dialogue systems with users with disabilities, a simulated environment was considered to be more suitable to evaluate the proposed POMDP framework. A framework that simulates users with different severities of dysarthria interacting with ASRs adapted with different amounts of data was developed. This framework permitted the comparison of different POMDP configurations in different scenarios. Even if the results are not as accurate as if they were evaluated with real users, they can be used as reference to investigate which of the POMDP configurations tested is more promising.

In summary, this chapter served as a first approach to analyse the suitability of using the POMDP-DM framework in VUIs personalised to dysarthric speakers. In addition, the conclusions obtained from this chapter serve as research questions for the following chapters. In the next chapter, a model-free policy optimization method is developed. This method is suitable to be used in the scenarios likely to be encountered in hS and, in addition, these scenarios are studied in more detail. Then, in chapter 5, slot-based dialogue state tracking models suitable for the hS scenarios are developed and evaluated.

Chapter 4

Personalised Reinforcement Learning based Policy Optimization

As the component that is in charge of the control of the dialogue flow, the policy model can be considered the core component of a dialogue manager. Historically, many systems have relied on rule-based models to tackle the Policy Optimization (PO) problem. However, due to their low adaptability, these models are not suitable for long-term personalised DM.

Data driven methods are a more appropriate approach for personalised PO due to their higher adaptability to different dialogue environments. Supervised learning methods have been studied for PO (Henderson et al., 2005, Meng et al., 2003), but these methods simply learn to "mimic" dialogue polices used by human experts. It is not clear if these policies are optimal, especially in noisy environments (Levin et al., 2000). The most promising data driven approach to PO is to model it as an RL problem under the POMDP framework (Williams and Young, 2007, Young et al., 2010, 2013). However, as it was reviewed in section 2, model-based RL approaches are intractable for large systems (Williams et al., 2005) and early proposed model-free RL-based approaches need an infeasibly large number of dialogues to be trained (Jurčíček et al., 2012). Recently, methods which need less data have been developed (Geist and Pietquin, 2010, Pietquin et al., 2011), Gaussian Process (GP)-based model-free RL Gašić and Young (2014) being one of the most promising. This approach has shown to reduce the number of dialogues needed to train a policy by various orders of magnitude (Gašić et al., 2011), making it very attractive for personalised PO.

However, these methods have only been studied in speaker independent scenarios, trained with data from several different speakers, while assuming that all the data comes from the same underlying distribution. It is also assumed that the environment is stable. Thus, data collected at any point in time will follow the same distribution. This is not the case in VUIs personalised to dysarthric speakers, as the speech characteristics of each speaker will be very different, possibly related to the type and severity of their dysarthria. If the acoustic models are adapted with speaker specific data as in hS, the ASR performance will also vary as more acoustic data is collected (Casanueva et al., 2014) and thus, the environment will vary. In the best case scenario, when large amounts of data coming from the "target" user and environment are available to train the policy model, there is no need for personalisation. However, this is rarely the case. For example, in a system such as homeService (Christensen et al., 2013b), data collection is very expensive because interacting through a prolonged period of time can be tiring for speakers with disabilities (Nicolao et al., 2016). It is much more likely to face a scenario where data from several "source" speakers (with different characteristics to the "target" speaker) is available. In addition, if the dialogue environment (e.g. the ASR) varies over time, there will be a mismatch between the data used to train the system and the data seen in the working conditions (Casanueva et al., 2014). All these issues lead to data collected from different (but related) distributions.

In these scenarios, the policy cannot assume that the data collected from different speakers, or with different ASRs, follows the same distribution. But it is also infeasible to collect enough data from the same user or ASR environment. Therefore, a trade-off between using data from different distributions while accounting on the differences between them is needed. In the context of speaker specific acoustic models for users with dysarthria, Christensen et al. (2014) demonstrated that using a speaker similarity metric to select the data to train the acoustic models for specific speakers improves ASR performance. Furthermore, using transfer learning methods (Taylor and Stone, 2009) to initialise the policy with data gathered from dialogue systems in different domains has been shown to increase the learning speed of the policy (Gašić et al., 2013) and provides an acceptable system performance when there is no domain-specific data available. In a project such as hS, several VUIs will be set up in different user's homes, letting the system developer collect data from different "source" speakers (Christensen et al., 2015). This can be considered analogous to having data from different domains. Joining these three ideas, when setting up a new system personalised to a "target" speaker, data gathered from the other "source" speakers can be used to pre-train the policy, defining a similarity metric between different speakers based on their speech characteristics. This metric can be used to select which data from the source speakers is used to train the model and to weight the influence of the data from each speaker in the model. As the user interacts with the system, source data can be discarded and the policy can be updated with the newly collected target data in an incremental way. In a scenario where the ASR varies over time, the data coming from interactions with different ASR performances follows different (but related) distributions. Thus, the same approach can be taken to select and weight the data if an ASR-performance metric is defined.

In summary, all the previously mentioned scenarios can be assumed to be Transfer Learning (TL) scenarios (Pan and Yang, 2010) where the data collected for training ("source" data) comes from distributions different to the actual "working condition" distribution (the distribution that needs to be modelled, the "target" data distribution) (Casanueva et al., 2014, Christensen et al., 2012a). If similarity metrics can be defined between the environments leading to the different distributions, the PO problem in these scenarios can be defined as a data selection and weighting problem where a distance based heuristic is used to select and weight the data. However, as mentioned in section 2, GP-RL soon becomes intractable as the data amount increases, limiting the amount of data from source speakers or environments that can be transferred[1]. Gašić and Young (2014) propose an approximated tractable method for GP-RL policy optimization, but this method does not allow to discard and select data.

Hence, to apply GP-RL in varying speaker and environment scenarios, approximation methods which allow data selection are needed. The main objective of this chapter is the development and evaluation of these methods. In addition, the methods should be able to adapt the models *online*: the methods should be able to update the models in a tractable way as new dialogue data is collected through interaction with the target user in the target environment.

This chapter presents several contributions: Firstly, two different GP-based RL methods, which allow data selection and discarding, are developed. The most promising method is based on the reformulation of the GP-RL equations presented in section 2.6.2 by defining a kernel in the *temporal difference* space. This not only leads to the possibility of application of several approximation methods used in GP regression, but also to the simplification of the equations for an easier understanding of GP-RL methods. Secondly, the new proposed Temporal Difference (TD) kernel approach is used to personalise dialogue policy models in two scenarios: in a scenario where the environment (ASR) changes over time and in a scenario where a policy for a target speaker is trained with data coming from

[1]It may also be the case, that due to the change of the target distribution, source data collected in the past which was useful to model previous target distributions is not useful any more and wants to be discarded

a set of source speakers with different dysarthria severities. Online adaptation of the models as target data becomes available through interaction is also studied. Finally, a set of environment and speaker features are defined. To select and discard the data, heuristics based on similarities between these features are used and the features are also integrated into the GP kernel to weight the data.

4.1 Tractable GP-RL based policy optimization for personalised VUIs

As mentioned in the previous section and in section 2.6.2, the main problem of using GP-RL models for dialogue PO is their computational complexity. Due to the inversion of a matrix of size $t \times t$ (the transformed Gram matrix), the complexity of computing the Q-function posterior (equation 2.30) in each turn is $\mathcal{O}(t^3)$ for each action, where t is the number of data points (the total number of turns in the case of dialogue management). As the number of interactions increase, this becomes quickly intractable, especially if the dialogue models are updated online.

One of the most popular GP approximation approaches is the definition of a representative set of inducing points, $\mathbf{U}_m$, which is used to approximate the inverted $t \times t$ matrix (Quiñonero-Candela and Rasmussen, 2005). In Engel et al. (2005) and Gašić and Young (2014), a GP-RL approximation method which reduces the complexity to $\mathcal{O}(t \times m^2)$ is proposed, where m is the size of the set of inducing points $\mathbf{U}_m$. However, the complexity reduction of this method is based on recursive updates of the inverted Gram matrix (Engel et al., 2003, 2004). This means that if a point needs to be discarded (e.g. because it was gathered a long time ago and the environment dynamics have changed), the whole GP would have to be retrained with a computational cost of $\mathcal{O}(t^3)$. In addition, the set of inducing points $\mathbf{U}$ is created dynamically as new data points are observed, so it is more difficult to control the size of the set of inducing points. To the knowledge of the author, this is the only approximation method proposed so far for model-free GP-RL models. However, in the GP regression literature, there are several approximation methods (sometimes called *sparsification* methods) which allow arbitrary data selection and discarding (Quiñonero-Candela and Rasmussen, 2005). These methods are not based on recursive updates, so it is therefore possible to recompute the inverted gram matrix in a tractable way every time new data becomes available. However, due to the differences between the equations of GP regression and GP-RL models, these approximation methods are not directly applicable to GP-RL.

The differences between the posterior equation in GP regression (equation 2.24) and GP-RL (equation 2.30) arise from the TD approach taken by the GP-RL models. This is reflected as the inclusion of the matrix $\mathbf{H}$, which defines the temporal difference relationship between two consecutive belief-action points. If methods to rewrite the GP-RL equation without the $\mathbf{H}$ operator are developed, sparsification methods designed for GP regression models would be straightforward to apply (Quiñonero-Candela and Rasmussen, 2005). Two of these methods are proposed in the following sections.

4.1.1 GP-MC approach

Following the *Bellman equation* (equation 2.12), the GP-RL approach models the Q-function using the observed immediate rewards and estimates of Q in the next state (equation 2.30). The $\mathbf{H}_t$ operator arises from this temporal difference relationship between two consecutive data points (dialogue turns). If, instead of a TD RL approach, a Monte-Carlo (MC) RL approach is taken (section 2.6.2), the GP can be modelled directly using the accumulated rewards c observed at the end of each dialogue. This approach follows the same idea as algorithm 2.2, with the difference that the Q-function estimates are not recursively updated.

Given a set of training dialogues, the set of visited belief-action points $\mathbf{X}_t$ (where $(\mathbf{x}_i = \mathbf{b}_i, a_i)$) with their observed accumulated rewards[1] $\mathbf{c}_t$ can be defined, where t is the number of points (number of turns for all dialogues). If each c_i is considered a random variable and is modelled dividing the Q-function in a mean and a residual function as:

$$c_i = Q(\mathbf{b}_i, a_i) + \Delta Q(\mathbf{b}_i, a_i) \tag{4.1}$$

given the set of observed belief-action points $\mathbf{X}_t$, with their respective $\mathbf{c}_t$ values observed at the end of each dialogue, the set of linear equations can be represented in matrix form as:

$$\mathbf{c}_t = \mathbf{q}_t + \Delta\mathbf{q}_t \tag{4.2}$$

where:

$$\begin{aligned}\mathbf{c}_t &= [c_1, c_2, ..., c_t]^\top \\ \mathbf{q}_t &= [Q(\mathbf{b}_1, a_1), ..., Q(\mathbf{b}_t, a_t)]^\top \\ \Delta\mathbf{q}_t &= [\Delta Q(\mathbf{b}_1, a_1), ..., \Delta Q(\mathbf{b}_t, a_t)]^\top\end{aligned}$$

If the set of random variables $\mathbf{q}_t$ is assumed to have a joint Gaussian distribu-

[1]The accumulated reward for each belief-action point is the reward accumulated from that point on the dialogue, not for the whole dialogue

tion with zero mean and $\Delta Q(\mathbf{b}_i, a_i) \sim \mathcal{N}(0, \sigma^2)$, the system can be modelled as a GP (Rasmussen, 2006). As in the GP-RL framework explained in chapter 2, the covariance matrix and vectors of the GP is determined by a *kernel function* defined independently over the belief and the action space (Engel et al., 2005):

$$k_{i,j} = k((\mathbf{b}_i, a_i), (\mathbf{b}_j, a_j)) = k^b(\mathbf{b}_i, \mathbf{b}_j)k^a(a_i, a_j) \tag{4.3}$$

From now on, $\mathbf{K}_{Y,Y'}$ is defined as the matrix of size $|\mathbf{Y}| \times |\mathbf{Y}'|$ whose elements are computed by the kernel function (equation 4.3) between the set of points $\mathbf{Y}$ and $\mathbf{Y}'$, where $\mathbf{Y}$ and $\mathbf{Y}'$ are any set of points. Using a MC RL approach, for any new belief-action point $\mathbf{x}_* = (\mathbf{b}_*, a_*)$, the posterior of the expected accumulated reward (The Q-function) can be computed using equation 4.2 as:

$$\begin{aligned}
&Q^{mc}(\mathbf{x}_*)|\mathbf{X}_t, \mathbf{c}_t \sim \mathcal{N}(\bar{Q}^{mc}(\mathbf{x}_*), \hat{Q}^{mc}(\mathbf{x}_*)) \\
&\bar{Q}^{mc}(\mathbf{x}_*) = \mathbf{K}_{*,X}(\mathbf{K}_{X,X} + \sigma^2\mathbf{I}_t)^{-1}\mathbf{c}_t \\
&\hat{Q}^{mc}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_{*,X}(\mathbf{K}_{X,X} + \sigma^2\mathbf{I}_t)^{-1}\mathbf{K}_{X,*}
\end{aligned} \tag{4.4}$$

where $\mathbf{X}_t$ is the set of size t of all the previously visited $(\mathbf{b}_i, a_i)$ points and $*$ denotes the set of size 1 composed by the new belief-action point to be evaluated. $\bar{Q}$ and $\hat{Q}$ represent the mean and the variance of Q respectively.

Using this approach, the Q posterior equation 4.4 has the same shape than in usual GP regression models, making the application of any approximation method proposed in the GP literature (Quiñonero-Candela and Rasmussen, 2005) straight forward. Furthermore, when the kernel used in the action space is the delta kernel (equation 2.29), the covariance between two belief-action points $\mathbf{x}_i$ and $\mathbf{x}_j$ where $a_i \neq a_j$ is always 0, which can be used to further decrease the computational complexity of the model.

In equation 4.4, when using the delta kernel (equation 2.29), the elements of the $\mathbf{K}_{*,X}$ vector corresponding to a point of the set $\mathbf{X}_t$ with a different action than a_* will be 0. If the set of training points is rearranged to group the points which have the same action together, the $(\mathbf{K}_{X,X} + \sigma^2\mathbf{I}_t)$ matrix will be block-diagonal. This means that only the belief-action points $\mathbf{x}_i$ where $a_i = a_*$ will affect the estimation of $Q^{mc}(\mathbf{b}_*, a_*)$. Taking this into account, an *action-GP* can be defined for each action $a \in \mathcal{A}$, defining the GP prior using only the subset of the points of $\mathbf{X}_t$ where $a_i = a$. Then, when computing the policy for each new belief point $\mathbf{b}_*$, the Q-function posterior for each action $a \in \mathcal{A}$ will be computed with the *action-GP* corresponding to that action, denoted as $Q^{mc}_a(\mathbf{b}_*, a)$. The active exploration policy (equation 2.20) can now be computed as:

$$\pi^a(\mathbf{b}_*) = \begin{cases} \arg\max_{a \in \mathcal{A}} \bar{Q}_a^{mc}(\mathbf{b}_*, a) \text{ with prob. } (1-\epsilon) \\ \arg\max_{a \in \mathcal{A}} \hat{Q}_a^{mc}(\mathbf{b}_*, a) \text{ with prob. } \epsilon \end{cases} \tag{4.5}$$

and the complexity is reduced to $\mathcal{O}(t_m^3)$, where $t_m = \max_{a \in \mathcal{A}}(t_a)$ and t_a is the number of points that have a as action. This is equivalent to defining an independent Q-function for each action, which can significantly reduce the complexity in a VUI where the size of the action set $\mathcal{A}$ is large. For online policy optimization, each *action-GP* prior can be updated after each dialogue (instead of after each turn, as it is possible with TD approaches).

4.1.2 Temporal Difference kernel

As mentioned in section 4.1.1, the $\mathbf{H}$ matrix in the GP-RL posterior function (equation 2.30) defines the TD relationship between two belief-action points consecutive in time. Due to this relationship, adjacent points (or dialogue turns) are co-dependent and cannot be discarded easily. The effect that this matrix has on the covariance matrices can be seen as performing a transformation of the covariance matrices in the belief-action space into the *temporal difference space*. This transformation is explained in more detail in the next paragraph.

In equation 2.30, a linear transformation from the belief-action space to the temporal difference space is applied to the covariance vector $\mathbf{K}_{*,X}$ and to the covariance matrix $\mathbf{K}_{X,X}$ by multiplying them by the matrix $\mathbf{H}_t$. Deriving the term $\mathbf{H}_t\mathbf{K}_{X,X}\mathbf{H}_t^\top$, the following matrix is obtained:

$$\mathbf{H}_t\mathbf{K}_{X,X}\mathbf{H}_t^\top =$$
$$\begin{bmatrix} \begin{matrix}(k_{1,1} + \gamma_1^2 k_{2,2} \\ -2\gamma_1 k_{1,2})\end{matrix} & \begin{matrix}(k_{1,2} + \gamma_1\gamma_2 k_{2,3} \\ -\gamma_2 k_{2,2} - \gamma_1 k_{1,3})\end{matrix} & \cdots & \begin{matrix}(k_{1,t-1} + \gamma_1\gamma_{t-1} k_{2,t} \\ -\gamma_{t-1} k_{2,t-1} - \gamma_1 k_{1,t})\end{matrix} \\ \begin{matrix}(k_{1,2} + \gamma_1\gamma_2 k_{2,3} \\ -\gamma_2 k_{2,2} - \gamma_1 k_{1,3})\end{matrix} & \vdots & \ddots & \vdots \\ \begin{matrix}(k_{1,t-1} + \gamma_1\gamma_{t-1} k_{2,t} \\ -\gamma_{t-1} k_{2,t-1} - \gamma_1 k_{1,t})\end{matrix} & \begin{matrix}(k_{2,t-1} + \gamma_2\gamma_{t-1} k_{3,t} \\ -\gamma_{t-1} k_{3,t-1} - \gamma_2 k_{2,t})\end{matrix} & \cdots & \begin{matrix}(k_{t-1,t-1} + \gamma_{t-1}^2 k_{t,t} \\ -2\gamma_{t-1} k_{t-1,t})\end{matrix} \end{bmatrix} \tag{4.6}$$

where $k_{i,j}$ is the kernel function between two belief-action points $\mathbf{x}_i = (\mathbf{b}_i, a_i)$ and $\mathbf{x}_j = (\mathbf{b}_j, a_j)$ defined in equation 4.3 and $\gamma_i = 0$ if a_i is a terminal action[1] or a

[1]In episodic tasks, such as dialogue management, the terminal actions are the machine actions that close an episode. In non-episodic RL tasks, γ will always be the normal discount factor.

discount factor $0 \leq \gamma \leq 1$ otherwise. The transformed matrix (equation 4.6) has the form of a covariance matrix where each element is a sum of kernel functions $k_{i,j}$ between belief-action points on time i or $i+1$ weighted by the discount factors. So each element ${k^{td}}_{i,j}$ of this matrix can be defined as a function of temporal differences between consecutive belief-action points, or as a function of two *temporal difference* points $\mathbf{z}_i = (\mathbf{b}_i, a_i, \mathbf{b}_{i+1}, a_{i+1}, \gamma_i)$ and $\mathbf{z}_j = (\mathbf{b}_j, a_j, \mathbf{b}_{j+1}, a_{j+1}, \gamma_j)$ as:

$$\begin{aligned} {k^{td}}_{i,j} &= k^{td}(\mathbf{z}_i, \mathbf{z}_j) \\ &= k^{td}((\mathbf{b}_i, a_i, \mathbf{b}_{i+1}, a_{i+1}, \gamma_i), (\mathbf{b}_j, a_j, \mathbf{b}_{j+1}, a_{j+1}, \gamma_j)) \\ &= (k_{i,j} + \gamma_i\gamma_j k_{i+1,j+1} - \gamma_i k_{i+1,j} - \gamma_j k_{i,j+1}) \end{aligned} \tag{4.7}$$

For the covariance vectors, if the term $\mathbf{K}_{*,X}\mathbf{H}_t^\top$ is derived, the following equation is obtained:

$$\begin{aligned} &\mathbf{K}_{*,X}\mathbf{H}_t^\top = \\ &\begin{bmatrix} (k_{1,*} - \gamma_1 k_{2,*}) & (k_{2,*} - \gamma_2 k_{3,*}) & \cdots & (k_{t-1,*} - \gamma_{t-1} k_{t,*}) \end{bmatrix} \end{aligned} \tag{4.8}$$

and deriving $\mathbf{H}_t\mathbf{K}_{X,*}$:

$$\begin{aligned} &\mathbf{H}_t\mathbf{K}_{X,*} = \\ &\begin{bmatrix} (k_{1,*} - \gamma_1 k_{2,*}) & (k_{2,*} - \gamma_2 k_{3,*}) & \cdots & (k_{t-1,*} - \gamma_{t-1} k_{t,*}) \end{bmatrix}^\top \\ &= \begin{bmatrix} \mathbf{K}_{*,X}\mathbf{H}_t^\top \end{bmatrix}^\top \end{aligned} \tag{4.9}$$

which are vectors with each element defined by ${k^{td}}_{i,*} = (k_{i,*} - \gamma_i k_{i+1,*})$. These elements can be computed with equation 4.7 by doing the following: redefining the new belief-action point being evaluated $\mathbf{x}_* = (\mathbf{b}_*, a_*)$ as a TD point as $\mathbf{z}_* = (\mathbf{b}_*, a_*, \mathbf{b}_{*+1}, a_{*+1}, \gamma_*)$, with $\gamma_* = 0$ and $\mathbf{b}_{*+1}$ and a_{*+1} set to any default value. Then, each element ${k^{td}}_{i,*}$ in 4.8 and 4.9 can be calculated with the function defined in equation 4.7 evaluated between the TD points $\mathbf{z}_i$ and $\mathbf{z}_*$.

Therefore, equation 4.7 is computing covariances between TD points. It is equivalent to a **kernel in the temporal difference space**. As this function is positive definite, satisfies the *Mercer condition* (Cortes and Vapnik, 1995), making it a valid kernel for GPs working in the TD space (Rasmussen, 2006). More formally, the set of belief-action points $\mathbf{X}_t$ is redefined as the set of TD points[1] $\mathbf{Z}_t$ where $\mathbf{z}_i = (\mathbf{b}_i, a_i, \mathbf{b}_{i+1}, a_{i+1}, \gamma_i)$, with $\gamma_i = 0$ and $\mathbf{b}_{i+1}$ and a_{i+1} set to any default value if a_i is a terminal action, or γ_i being a normal discount factor $1 \leq \gamma_i \leq 0$ otherwise.

[1]Take into account that, even if it is not included in the notation, $|\mathbf{Z}_t| = |\mathbf{X}_t| - 1$, as the set of TD points is composed by all the consecutive pairs of belief-action points.

Defining

$$\begin{aligned}\mathbf{K}^{td}_{*,Z} &\equiv \mathbf{K}_{*,X}\mathbf{H}_t^\top \\ \mathbf{K}^{td}_{Z,*} &\equiv \mathbf{H}_t\mathbf{K}_{X,*}\end{aligned} \tag{4.10}$$

as the covariance vectors between the set of TD points and the new TD point $\mathbf{z}_*$ being evaluated computed with the TD kernel (equation 4.7) and defining

$$\mathbf{K}^{td}_{Z,Z} \equiv \mathbf{H}_t\mathbf{K}_{X,X}\mathbf{H}_t^\top \tag{4.11}$$

as the covariance matrix or *Gram matrix* between the set of observed TD points computed with the TD kernel, it is possible to rewrite the Q-function posterior (equation 2.30) for any new TD point as:

$$\begin{aligned}&Q^{td}(\mathbf{z}_*)|\mathbf{Z}_t, r_{t-1} \sim \mathcal{N}(\bar{Q}^{td}(\mathbf{z}_*), \hat{Q}^{td}(\mathbf{z}_*)) \\ &\bar{Q}^{td}(\mathbf{z}_*) = \mathbf{K}^{td}_{*,Z}(\mathbf{K}^{td}_{Z,Z} + \mathbf{\Sigma}_t)^{-1} r_{t-1} \\ &\hat{Q}^{td}(\mathbf{z}_*) = k^{td}(\mathbf{z}_*, \mathbf{z}_*) - \mathbf{K}^{td}_{*,Z}(\mathbf{K}^{td}_{Z,Z} + \mathbf{\Sigma}_t)^{-1}\mathbf{K}^{td}_{Z,*}\end{aligned} \tag{4.12}$$

With this notation, as the transformation to the TD space is performed by the kernel, the operator $\mathbf{H}_t$ disappears (except from the noise term $\mathbf{\Sigma}_t$), simplifying the notation. Furthermore, the redefinition of the set of points and the kernel has several advantages:

- Redefining the belief-action set of points $\mathbf{X}_t$ as the set of temporal difference points $\mathbf{Z}_t$ simplifies the selection and discarding of data points, because the dependency between consecutive points is well defined. Therefore, TD points are independent of each other and can be discarded without breaking any dependency.

- If it is assumed that $\Delta Q(\mathbf{b}_i, a_i) - \gamma_i \Delta Q(\mathbf{b}_{i+1}, a_{i+1}) \sim \mathcal{N}(0, \sigma^2)$ (as proposed in Engel et al. (2003)), the noise term becomes $\mathbf{\Sigma}_t = \sigma^2\mathbf{I}$. Thereafter, the shape of the equation for the posterior of Q^{td} is the same than the classic GP regression models (equation 2.24). Thus, it is straightforward to apply a wide range of well studied GP techniques, such as sparsification methods.

As mentioned in this chapter, the main problem with the approximation method for GP-RL proposed by Engel et al. (2005) is that the arbitrary selection and discarding of data points is not possible. On the other hand, the GP literature proposes several approximation methods[1] which select a subset of *inducing points* $\mathbf{U}_m$

[1] A review of these methods can be found in Quiñonero-Candela and Rasmussen (2005)

of size $m < t$ from the set of training points $\mathbf{Z}_t$. Due to the shape of equation 4.12, obtained by using the TD kernel, these methods are straight forward to apply. Using one of these methods, heuristic approaches[1] can be used to select the set $\mathbf{Z}_t$ and $\mathbf{U}_m$ in each turn, making the online update of the policy model tractable. In the following experiments, the Deterministic Training Conditional (DTC) (Seeger et al., 2003) method is used. With this approximation method, once the subset of points has been selected in each turn, the Q-function posterior can be approximated with $\mathcal{O}(t \cdot m^2)$ complexity as:

$$\begin{aligned} Q^{dtc}(\mathbf{z}_*)|\mathbf{Z}_t, r_{t-1} &\sim \mathcal{N}(\bar{Q}^{dtc}(\mathbf{z}_*), \hat{Q}^{dtc}(\mathbf{z}_*)) \\ \bar{Q}^{dtc}(\mathbf{z}_*) &= \sigma^{-2}\mathbf{K}^{td}_{*,U}\mathbf{\Lambda}\mathbf{K}^{td}_{U,Z}r_{t-1} \\ \hat{Q}^{dtc}(\mathbf{z}_*) &= k^{td}(\mathbf{z}_*, \mathbf{z}_*) - \mathbf{\Phi} + \mathbf{K}^{td}_{*,U}\mathbf{\Lambda}\mathbf{K}^{td}_{U,*} \end{aligned} \tag{4.13}$$

where $\mathbf{\Lambda} = (\sigma^{-2}\mathbf{K}^{td}_{U,Z}\mathbf{K}^{td}_{Z,U} + \mathbf{K}^{td}_{U,U})^{-1}$ and $\mathbf{\Phi} = \mathbf{K}^{td}_{*,U}(\mathbf{K}^{td}_{U,U})^{-1}\mathbf{K}^{td}_{U,*}$ (refer to Quiñonero-Candela and Rasmussen (2005) for a more detailed explanation).

Once the posterior for any new belief-action point can be computed with equation 4.12 or equation 4.13, the policy $\pi(\mathbf{b}) = a$ can be computed following an *ϵ-greedy* active exploration approach (equation 2.20).

4.2 Transfer Learning between speakers for GP-RL

The introduction of this chapter described some scenarios that can be found when setting up a personalised VUI, one of them being the scenario where a dialogue manager has to be trained for a *target* speaker using data from several different *source* speakers. If the speech characteristics of the speakers are very different (e.g. dysarthric speakers with different severities of dysarthria), the data samples gathered from each source speaker will have been drawn from a different distribution, which at the same time will be different to the target speaker distribution (the distribution that needs to be modelled). This scenario, where a statistical model has to be trained for a specific "target" task, but only data from different but related "source" tasks is available, is known as transfer learning (Pan and Yang, 2010). In reinforcement learning, TL has been shown to increase the performance of the system in the initial stages of use (called *jumpstart*) and to speed up policy learning (Taylor and Stone, 2009). In the context of TL for dialogue policy optimization in different domains, Gašić et al. (2013) showed a significant reduction on the amount of target dialogues needed to reach the optimal policy when the model was ini-

[1]The proposed data selection heuristics are explained in section 4.2

tialised with dialogues from other domains. In the context of a personalised DM trained with data from different speakers, dialogue interactions with each speaker can be considered different tasks, as the data collected from each speaker will be coming from a different distribution. The TD kernel approach developed in section 4.1.2 combined with the DTC approximation method gives an excellent framework for TL in policy optimisation, as it simplifies the data selection. In this section, three aspects of applying TL to personalised PO are addressed:

- *How to transfer the knowledge*: Which transfer learning models are the most appropriate for personalised policy optimisation?

- *Which data to transfer*: Which methods or metrics can be used to select the most relevant data?

- In the case of multiple source speakers, *How to weight the relevance of the different sources*: Which methods or metrics have to be used to weight the data in order to account for the relevance of different sources on the target model?

4.2.1 Knowledge transfer

Several TL approaches to transfer knowledge from source tasks to a target task exist (Pan and Yang, 2010), each of them performing better in different situations. This section addresses which is the most appropriate model to transfer the data from the source speakers to initialise the policy for the target speaker. The most straightforward way to transfer the data in GP-RL is to initialise the set of temporal difference points $\mathbf{Z_t}$ of the GP with the source points (the data points coming from the source speakers), and then continue updating it with target data points as they are gathered through interaction[1]. However, this approach has a few shortcomings. Firstly, as the complexity of GP-RL models increases with the number of data points, the model might quickly become intractable if it is initialised with too many source points. Also, after data points from the target speaker are gathered through interaction, the source points may not improve the performance of the system, while increasing the model complexity. Secondly, as the computation of the Q-function posterior variance for a new point ($\hat{Q}$ in equation 4.13) depends on the number of similar points already visited, the variance of the new belief-action points being evaluated will be reduced by the effect of the source points close in the belief-action space. If the distribution of the source data points is unbalanced,

[1]If a sparse approach such as DTC is used, the set of inducing points will also be initialised with the source data, but this issue is addressed in the next subsection (Transfer data selection).

the effectiveness of the active exploration policy (equation 2.20) will be affected, because the variance in the areas of the belief-action space with a high density of source points will be small and thus, these parts of the space will never be explored.

To deal with the variance computation problem, Gašić et al. (2013) proposes to use the source points to train a *prior GP*[1] and use the Q posterior of this GP as the mean function of the GP trained with the target points (instead of using a zero-mean GP). With this approach, the mean of the posterior in equation 4.12 is modified as:

$$\bar{Q}(\mathbf{z}_*) = m(\mathbf{z}_*) + \mathbf{K}^{td}_{*,Z}(\mathbf{K}^{td}_{Z,Z} + \Sigma)^{-1}(r_{t-1} - \mathbf{m}_t) \tag{4.14}$$

where $m(\mathbf{z}_*)$ is the mean of the posterior of the Q-function given by the prior GP and $\mathbf{m}_t = [m(\mathbf{z}_0), ..., m(\mathbf{z}_t)]^\top$. If the DTC approach (equation 4.13) is taken, the posterior Q-function mean becomes:

$$\bar{Q}^{dtc}(\mathbf{z}_*) = m(\mathbf{z}_*) + \sigma^{-2}\mathbf{K}^{td}_{*,U}\mathbf{\Lambda}\mathbf{K}^{td}_{U,Z}(r_{t-1} - \mathbf{m}_t) \tag{4.15}$$

In other words, this approach uses the mean of the posterior, $\bar{Q}$, of a GP trained with the source points as a "prior distribution" for the GP trained with the target points. This way, when there are no target data points available, the model uses the prior GP trained with source points to estimate the Q-function. As target data points become available, these points will "correct" the prediction given by the prior GP, until there is enough target data so that the effect of the target points overcomes the effect of the prior GP in the model[2]. This approach has the advantage of being computationally cheaper than the former method while modelling the uncertainty for new target points more accurately, but at the cost of not taking into account the correlation between source and target points. This might reduce the performance when the amount of target data is small.

A third approach is to combine the two previous methods, using a portion of the transfer points to train a GP for the prior mean function, while the rest is used to initialise the set $\mathbf{Z}_t$ of the GP that will be updated with target points. This method is computationally cheaper than initialising $\mathbf{Z_t}$ with all the source points, and at the same time increases the performance of the method using equation 4.15 when the amount of target data is small.

[1]Do not confuse with the *GP-prior*

[2]The effect of the mean of a GP is decreased as the number of data points increases (Rasmussen, 2006). This is why choosing a zero-mean GP does not affect the performance while simplifying the computation.

4.2.2 Transfer data selection

Because GPs are non-parametric models, their complexity increases with the number of training data points, limiting the amount of source data that can be transferred. Additionally, if the points come from multiple sources, it is possible that the data distribution from some sources is more similar to the target speaker than others, hence transferring data from these sources will increase performance. If a speaker feature vector[1] $\mathbf{s} \in \mathbb{R}^n$ can be extracted from each speaker in a metric space where a similarity function $f(\mathbf{s}, \mathbf{s}')$ between speakers can be defined, heuristic methods can be used to select which data to transfer based on this similarity. When setting up a new system for a target speaker, the most straight forward heuristic for data selection is to transfer the data from the most similar speakers up to a maximum amount. More advanced selection methods can also be used, such as heuristics that trade off between selecting the most similar speakers and covering the belief-action space as uniformly as possible.

When the system is set up and the target speakers begins interacting with it, target data points will be collected. These points follow the distribution that has to be modelled, hence they should be included in the model while source points are discarded. The DTC approximation method (equation 4.13) allows this to be done in an online fashion while maintaining a tractable model complexity. In addition, when using the DTC approach, a subset of inducing points $\mathbf{U}_m$ must be selected. The most straightforward way is to select the subset of points most similar to the target speaker within the transferred points. As the user interacts with the system and target data points are gathered, these points may be used as inducing points. This approach acts like another layer of data selection; the reduced complexity will allow for the transfer of more source points, whereas using the target points as inducing points will mean that only the source points that lie in the same part of the belief-action space as the target points have influence on the model.

4.2.3 Transfer data weighting

When the transferred source data comes from multiple speakers, the similarity between each source speaker and the target speaker might be different. For example, if the target speaker has mild dysarthria, mild dysarthric source speakers will be more similar to the target than severe dysarthric ones (Christensen et al., 2014). Thus the data from a source more similar to the target should have more influence in the model than less similar ones. The similarities that can be computed with

[1]These can also be called *environment vectors*, as they give information about some part of the environment (in this case the speaker)

the speaker feature vectors $\mathbf{s}$ proposed in section 4.2.2 can also be used to weight the data.

In a GP-RL based policy, the kernel can be used to weight the data coming from different speakers (or environments). The kernel in a GP computes the covariance between data points, which is a kind of similarity. A large covariance between two points means that the two points are similar and a small covariance means that they are different. If the kernel in the *belief-action* space defined in equation 4.3 is extended into a kernel in the *belief-action-environment* space, k^e, by factorising the kernels in the belief, action and environment space as:

$$\begin{aligned} k^e_{i,j} &= k((\mathbf{b}_i, a_i, \mathbf{s}_i), (\mathbf{b}_j, a_j, \mathbf{s}_j)) \\ &= k^b(\mathbf{b}_i, \mathbf{b}_j) k^a(a_i, a_j) k^s(\mathbf{s}_i, \mathbf{s}_j) \end{aligned} \tag{4.16}$$

where $k^s(\mathbf{s}_i, \mathbf{s}_j)$ is any kind of *Mercer kernel* (Schölkopf and Smola, 2002), the covariance values used to compute the Q-function posterior will be also influenced by the similarity between points in the *speaker space* (or *environment space*). Therefore, points gathered with source speakers which are more similar to the target one will have more influence in the computation of the Q-function. This approach also helps to partially deal with the variance computing problem of the first transfer learning model in section 4.2.1, as the source points will lie on a different part of the speaker space than the new target points, consequently having less influence in the variance computation.

Note that the extended kernel is the kernel in the belief-action space. As the TD kernel defined in equation 4.7 is a function of the kernel in the belief-action space, after the kernel extension proposed in this section, the TD kernel becomes a function of the kernel in the *belief-action-speaker* space (or *belief-action-environment* space). The set of TD points is then redefined as $\mathbf{z}_i = (\mathbf{b}_i, a_i, \mathbf{s}_i, \mathbf{b}_{i+1}, a_{i+1}, \mathbf{s}_{i+1}, \gamma_i)$.

4.2.4 Speaker similarities

To compute the similarities between speakers, a way to extract a speaker feature vector $\mathbf{s}$ from each speaker must be chosen. The chosen feature vectors should be based on measurable characteristics of the speakers or the ASR-SLU channel which have an effect on the environment from the dialogue managers point of view. In this section three different approaches to extract these features are proposed:

- **Intelligibility assessment**: Figure 3.3 showed that the ASR performance is directly related to the dysarthria severity (or intelligibility) of the speaker. The intelligibility of dysarthric speakers can be assessed by testing the percentage of words understood by unfamiliar listeners (Kim et al., 2008). The numeric value corresponding to the intelligibility of each speaker can be used as a single dimensional speaker feature **s**.

- **i-vectors**: Martínez et al. (2013) showed that *i-vectors* (Dehak et al., 2011) can be used to predict the intelligibility of a dysarthric speaker. Therefore, the *i-vectors* are a potentially useful feature to compute similarities between dysarthric speakers without the need to perform intelligibility assessments.

- **ASR accuracy per command:** Ultimately, the effect of the full VUI environment (speaker, ASR and SLU) is reflected in the output of the SLU (or of the ASR in homeService). A measure of the confusability of the ASR or SLU output can be a useful feature to estimate similarities between different speakers (or environments). In a small sized VUI such as hS, if a held out set of recordings from the user is available, this set can be used to estimate the average ASR accuracy for each word in the vocabulary. Using these estimates, **s** can be defined as the vector with the size of the ASR vocabulary (36 in the following experiments) where each element is the average ASR accuracy of each command.

4.3 Policy Optimization in variable environments

Most of the published research on dialogue policy optimization assumes that all the data comes from the same distribution. In an agent-environment RL interaction, this assumption implies that the environment dynamics are stable, they do not change over time. This is not the case in a personal VUI where the ASR performance improves as the user interacts with the system and acoustic data is gathered (as it happens in hS). Interacting with different environments can be considered as different tasks, therefore, the same TL principles analysed in the previous section can also be applied in this scenario.

In systems like hS, however, the environment does not change randomly. Observing figure 3.3, it can be seen how changes of the ASR performance as the amount of user data increases are smooth. Imagine a system whose ASR is updated every time a minute of acoustic data is gathered. Every time the ASR is updated, the agent (DM) will be interacting with a different environment. Thus, the dialogue points gathered will come from different probability distributions. Lets call these distributions $ASR_0, ASR_1, ASR_2, ASR_3...$, where the sub-index indicates the amount of minutes of acoustic data used to adapt the models. Then, based on the ASR accuracy behaviour observed in figure 3.3, it can be assumed that the distribution ASR_3 is more similar to the distribution ASR_2 than it is to ASR_1 and so on.

Based on this assumption, each data point can be assigned with an *environment feature* vector $\mathbf{s}$ related to the *ASR environment* where it was collected (e.g. the amount of data used to train the ASR). These environment feature vectors can be then used to weight and select the data coming from different *ASR environments*, using the same approaches described in sections 4.2.2 and 4.2.3.

4.3.1 Environment metrics

To apply the TL approaches described in section 4.2 to this scenario, a numeric value $\mathbf{s} \in \mathbb{R}^n$ must be assigned to each *ASR environment* in a metric space where similarities between environments can be effectively computed. Depending on the nature of the changes in the environment, this can be a non-trivial task. However, when the changes in the environment are related to the ASR performance, there are simple ways to define the "environment vectors" $\mathbf{s}$. Two different ways to define the environment features are proposed:

- **User specific adaptation data amount:** As observed in figure 3.3, the ASR performance is directly related to the amount of user specific data used to adapt the acoustic models. Therefore, the number of single word recordings used to adapt the ASR can be used as a single-dimensional environment feature.

- **ASR accuracy per word:** As explained in section 4.3, in a small sized VUI a held out set of recordings from the user can be used to estimate the average ASR accuracy for each word in the vocabulary. Therefore, $\mathbf{s}$ can be defined as the vector with the size of the ASR vocabulary where each element is the average ASR accuracy of each command.

4.4 Transfer Learning experiments in the homeService environment

To evaluate the techniques presented in this chapter, a set of experiments is performed in the simulated hS environment presented in section 3.2.3. This is a scenario with high variability between the dynamics of the speakers and where an environment that varies over time can be simulated. Therefore, it is an appropriate scenario for testing the approaches described in this chapter.

In this section, three different scenarios are evaluated, each one trying to address the effectiveness of the proposed methods in specific situations that are likely to occur in personalised VUIs. In the first scenario, the two approximate models presented in section 4.1 are trained from scratch and their performance is compared with a non-approximated GP-RL model. The objective of these experiments is to identify the most appropriate approximate model for policy optimisation in VUIs. In the second scenario, an environment where the ASR performance improves over time is simulated, with the objective of testing techniques proposed in section 4.3. Thirdly, to test the data transfer, selection and weighting techniques proposed in section 4.2, a transfer learning scenario between speakers is simulated; for each test speaker, a DM is initialised with the data of the other 14 speakers, and then the DM is updated online as data from the test speaker becomes available through interaction.

4.4.1 Experimental set-up

The system is organised in the tree setup described in section 3.2.2 and the same evaluation metrics are used (-1 for each dialogue turn, +20 if the dialogue is successful). The state tracker is a logistic regression classifier (Metallinou et al., 2013), where classes are the set of states $\mathcal{S}$. The belief state $\mathbf{b}$ is computed as the posterior over the states given the last 5 observations (N-best lists with normalised confidence scores). For each speaker, the state tracker has been trained with data from the other 14 speakers.

GP models have several *hyperparameters* (e.g. Gaussian noise variance or kernel parameters) that need to be tuned. Even if these hyperparameters can be automatically optimised (Chen et al., 2015), most of the published work in GP-RL obtained better results by hand-tuning the parameters (Engel et al., 2005, Gašić and Young, 2014). In the following experiments, grid optimization methods are used to tune the hyperparameters. Following this approach, the Gaussian noise variance σ^2 in the GP-RL model (equation 4.12) is set to 5. The kernel used over the action space is the delta kernel (equation 2.29) and the kernel used over the belief space is a Radial Basis Function (RBF) kernel (a.k.a. Gaussian Kernel) :

$$k^b(\mathbf{b}_i, \mathbf{b}_j) = \sigma_k^2 \exp\left(- \frac{||\mathbf{b}_i - \mathbf{b}_j||^2}{2l_k^2} \right) \tag{4.17}$$

with *kernel variance* $\sigma_k^2 = 25$ and *lengthscale* $l_k^2 = 0.5$.

To remove the variability introduced by the random goal choice of the simulated user, each experiment has been initialised with five different seeds and tested over 500 dialogues for each number plotted. In each set of experiments, the system is tested with six different simulated users trained with data from low and medium intelligibility speakers[1]. This is because, as shown in chapter 3, a 36 command setup statistical DM is most useful for low and medium intelligibility speakers. For high intelligibility speakers, the ASR accuracy is close to 100%, so the improvement obtained from DM is small. For very low intelligibility speakers, the absolute performance is not high enough to make the system useful. These six speakers will be called the *test speakers* from now on.

4.4.2 Comparing different approximate models

In this section, the two tractable GP-RL models introduced in section 4.1 (named *GP-MC* and *DTC-Sarsa*) and the original GP-RL model proposed in Gašić and Young (2014) (*GP-Sarsa*) are trained from scratch, with the objective of comparing their policy learning speed and performance. The three models compared and their respective parameter configuration are the following:

- *GP-MC*: This approach computes the Q-function with the Monte-Carlo approach described in equation 4.4 and follows the policy defined in equation 4.5. The Gaussian noise variance and the kernel function with fixed hyperparameters described in section 4.4.1 are used. The maximum number of points to train each GP prior is defined 500 per action.

- *GP-Sarsa*: This policy, corresponding to the non-approximated GP-RL model, is used as an upper bound to evaluate the approximation methods. It follows the active exploration policy approach defined in equation 2.20 computing the Q-function with equation 4.12. The kernel function k_{ij} inside the TD kernel (equation 4.7) is the same kernel function used for the *GP-MC* policy. The maximum number of points used to train the GP prior is 1200[2].

- *DTC-Sarsa*: This policy uses the combination of the TD kernel approach with the application of the DTC approximation methods to compute the Q-function following equation 4.13. It uses an active exploration policy approach, following equation 2.20. The *TD-kernel* is the same as in GP-Sarsa. The maximum number of points is 1200 and the size of the inducing set U is 300.

[1]The data used to train the models in the TL scenario still comes from all the 15 dysarthric simulated users.

[2]The maximum number of points for *GP-MC* and *GP-Sarsa* is set up differently because *GP-MC* defines a GP for each action and thus can accept a larger number of (total) points. However, in the following experiments the models are trained from scratch; hence, the maximum number of points has very little effect on the results.

Experimental results

Fig. 4.1 shows the performance of the six different test speakers interacting with a total of six ASRs adapted with different amounts of data *(0, 50, 100, 150, 300 and 500 words)*, making a total of 36 speaker-environment pairs. The results presented are the average dialogue success rate and reward of the 36 pairs, for the three evaluated policy models (*GP-MC*, *GP-Sarsa* and *DTC-Sarsa*). The x axis is the number of dialogues used to train the policy. *GP-Sarsa* shows the best performance, which is the expected result taking into account that it is the non-approximated method used as upper bound. The success rate of *DTC-Sarsa*, however, is only slightly lower for any amount of training dialogues. The reward of *DTC-Sarsa*, however, decreases with respect to *GP-Sarsa* as the amount of training dialogues increases, showing that the policy is taking more turns to complete the dialogues than *GP-Sarsa*. *GP-MC*, in contrast, has a lower success when the amount of training dialogues is small but slightly outperforms the other two policies when the amount of training dialogues increases. In the reward metric, one can observe that *GP-MC* has a slower learning rate than *DTC-Sarsa*, even if after 500 training dialogues the reward of *GP-MC* is similar to *DTC-Sarsa*. To observe the performance of the

Policy	Reward	Success
GP-Sarsa	10.65	79.82%
GP-MC	10.19	88.90%
DTC-Sarsa	10.47	74.97%

Table 4.1: *Results for different policies trained until convergence*

models when the success rate and reward have converged, the three policy models were trained until convergence. To do so, each node (sub-dialogue) of the tree setup structure was trained independently with 900 sub-dialogues[1]. Table 4.1 shows the performance of the three models trained until convergence. Once again, *GP-Sarsa* obtains the best reward and success results. *DTC-Sarsa* obtains almost the same reward, but a success rate 5% lower, showing that the tendency seen in figure 4.1 is reverted and the policy is favouring shorter dialogues. On the contrary, *GP-MC* obtains a reward 0.5 lower than the other two models, but has a success rate 9% higher. This shows that this policy is favouring dialogues considerably longer than the other two.

[1]Take into account that this is not equivalent to training the system with 900 dialogues, since the tree structure of the dialogue state slows down the policy learning of the nodes deeper in the tree. Training each node independently deals with this problem.

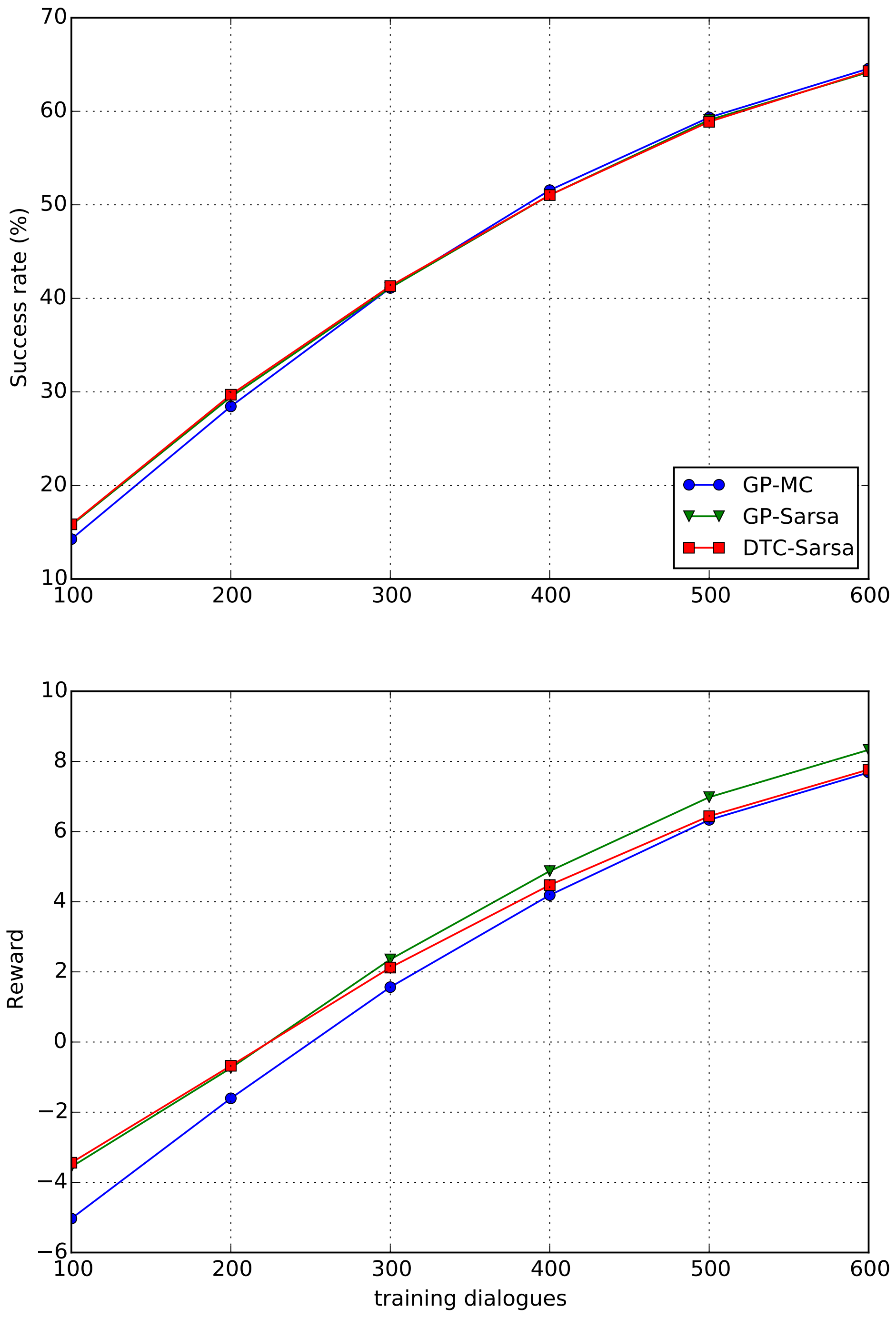

Figure 4.1: *Performance of GP-Sarsa (green), GP-MC (blue) and DTC-Sarsa (red). The x axis represents the number of training dialogues.*

Experiment conclusions

The objective of these experiments was to identify which policy approximation model, *GP-MC* or *DTC-Sarsa*, is more suitable for a personalised VUI. In the experiments it is shown how each model has its advantages and disadvantages. When the number of training dialogues is small, *GP-MC* shows a lower learning rate than *DTC-Sarsa*. When the models are trained until convergence, *GP-MC* showed a higher success rate but a lower reward, meaning that the *GP-MC* policy is leading to longer but more successful dialogues. The reason for this might be the following: as MC-RL methods update the Q-function estimate of each visited belief-action point after the dialogue is completed, the method cannot distinguish which of the visited points leaded to a successful or failed dialogue. Therefore, *GP-MC* methods rely on successful dialogues to obtain good estimates of the expected accumulated reward, so the policy learning speed is reduced and the policy tends to give more importance to the success rate over dialogue length. Even if this can be useful in scenarios where large amounts of data are available, this is not the case in personalised VUIs, where the models need to be adapted quickly to the user and to the changes in the environment. TD-based RL methods, in contrast, do not have this problem, because the Q-function estimate is updated using the local reward and the estimate of the Q-function in the next point, thus speeding up the policy learning. In addition, dialogue length is usually more important than success rate in home control VUIs, where an unsuccessful dialogue (e.g. changing the channel to 5 instead of to 4) is not a substantial problem, whereas taking several turns to change the TV channel might be annoying for the user. In any case, the ratio between success rate and dialogue length can be modified by tuning the reward function. Overall, the results suggest that, compared to *GP-MC*, *DTC-Sarsa* is a better GP-RL approximation approach for policy optimisation of personalised VUIs, having a performance comparable to *DTC-Sarsa* with a much smaller computational complexity.

4.4.3 Policy Optimization in variable environments

The objective of the next set of experiments is to evaluate the performance of the *DTC-Sarsa* policy model in a varying environment scenario, where the ASR performance improves as it is adapted with speaker specific data. To do so, the approach described in section 4.3 will be used. The description of the scenario is the following: each test speaker interacts with a set of ASR simulators, adapted with an increasing amount of acoustic data (from 0 adaptation words to 350, with an increase of 50 words each step). For each step, each *speaker-ASR* pair interacts with a

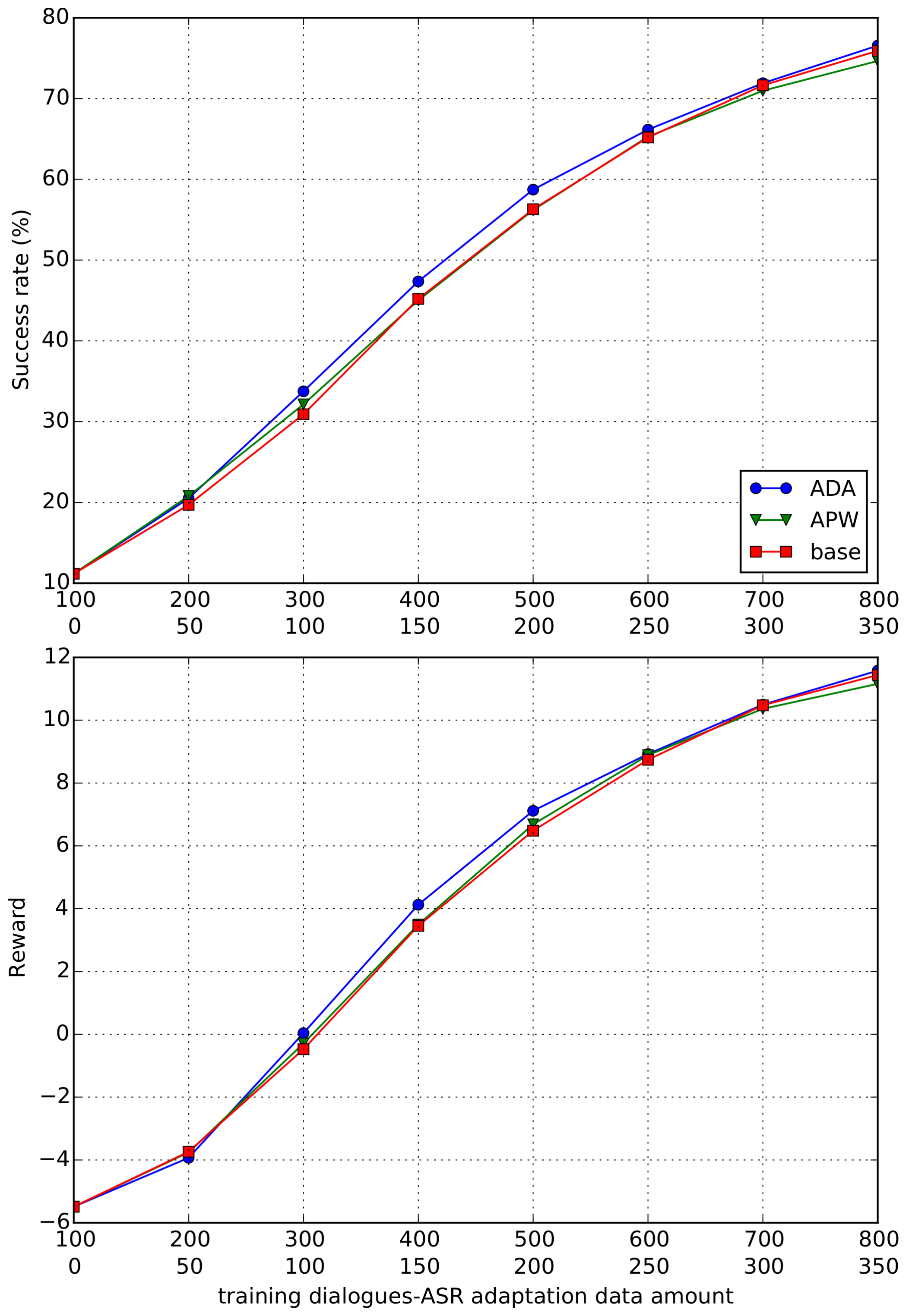

Figure 4.2: *Performance of DTC-Sarsa in a variable environment for different environment features. The x axis represents the number of training dialogues and the number of words used to adapt the ASR.*

dialogue manager trained with 100 dialogues collected in that environment (ASR), plus 100 dialogues collected in each of the previous environments. For example, in the first step, the simulated user will interact with an ASR simulator adapted with 0 words and with a policy trained with 100 dialogues collected in this ASR environment. In the second step, the user will interact with an ASR simulator adapted with 50 words and with a policy trained with 100 dialogues collected in the 50-word ASR environment and 100 dialogues collected in the 0-word ASR environment. In the next one, the user will interact with an ASR adapted with 100 words and the policy trained with 300 dialogues, 100 collected in the current environment (100-word), and 200 collected in the 0-word and 50-word environments, and so on. This setting tries to simulate a scenario representing varying environment dynamics, where the ASR is improving over time and the training data has been collected interacting with different ASRs.

The *DTC-Sarsa* policy model uses the same hyperparameter setting than in section 4.4.2. To analyse which environment features **s** are more useful, the model is tested using the features proposed in section 4.3: Adaptation data amount (ADA) and Accuracy per word (APW). The policy using these features is also compared to a baseline that does not use any environment feature (*base*). The kernel over the environment space k^s is also a RBF kernel (equation 4.17) with variance $\sigma_b = 1$. The kernel lengthscale is $l_b = 500$ for the *ADA* features and $l_b = 4$ for the *APW* features.

Experimental results

Figure 4.2 shows the mean dialogue performance of the six test speakers in the varying environment scenario. The x axis shows two values: the amount of dialogues used to train the policy (up) and the amount of data used to adapt the ASR (down). The y axis represents the dialogue success rate in the top figure and the dialogue reward in the bottom one.

When the *ADA* features are used, three different ranges of behaviour can be observed. In the first range, corresponding to the ASR environments from *0* to *50*, the performance of these features is very similar to the baseline. In the second range, corresponding to the environments from *100* to *250*, using *ADA* features gives an improvement of up to 0.7 in reward and 2% in success with respect to the baseline. In the third range, from *300* to *350*, the performance is again very similar to the baseline. Note how the improvement respect to the baseline increases as the amount of adaptation data increases, reaching its maximum at the *200* environment and then decreasing again. Using the *APW* features also give a slight improvement with respect to the baseline, but this improvement is much smaller than the one obtained with the *ADA* features.

Experiment conclusions

The objective of these experiments was to analyse the performance of the approach proposed in section 4.3 in a varying environment scenario, using two different environment features. The proposed method using *ADA* features gave a statistically significant improvement in the range of ASR environments going from 100 to 250. Observing the ASR behaviour in figure 3.3, it can be seen that the range of environments in where *ADA* features gives the highest improvement is also the range of environments that presents the highest variation between their dynamics. Therefore, the proposed method performs best when the differences between the dynamics are varying more, which is a promising result. The reason for the small improvement in the first environments (from 0 to 100) might be that the model does not have enough training data to make environment features useful, while in the later environments (250 to 350) the ASR improvement has already converged so the variation between environments is very small.

Lastly, note how the improvement obtained by *APW* is much smaller than the one obtained by *ADA* features. This is surprising because *APW* features should give a better estimate of the similarities between environments, as they directly account for the ASR performance. Research on more useful environment features could improve the performance of this method.

4.4.4 Transfer Learning between speakers

The objective of the experiments presented in this section is to explore the effect of applying the TL techniques proposed in section 4.2 in a hS environment personalised to a speaker. To test the effect of training a personalised policy model transferring knowledge from other speakers, for each test speaker (the target speaker), the policy models are initialised with data from the other 14 source speakers. Then, to test if the policy can be adapted online as the user interacts with the system, increasing amounts of target speaker dialogues are included in the training data. Each test speaker is evaluated in different ASR environments, interacting with three different ASRs adapted with 0, 150 and 300 user specific words. The main objective of the experiments is to compare the performance of the different models, speaker features and data selection heuristics proposed in section 4.2. To do so, the three different TL models proposed in section 4.2.1 are compared:

- *DTC*: Equation 4.13 is used to compute the Q posterior for the active exploration policy (equation 2.20) and the set of temporal difference points $\mathbf{Z}_t$ is initialised with the transferred source points.

- *Prior*: Equation 4.15 is used to compute the Q posterior for the active exploration policy. The prior GP to compute the mean function ($m(z_*)$ in equation 4.15) is trained with the transferred source points.
- *Hybrid*: Equation 4.15 is used to compute the Q posterior for the active exploration policy. The prior GP is trained with half of the source points and the set of temporal difference points $\mathbf{Z}_t$ is initialised with the other half.

as well as the two different data selection methods proposed in section 4.2.2

- *Close*: The data points from the source speakers closest to the target speaker in the speaker space are selected to be transferred. The set of inducing points $\mathbf{U}_m$ is initialised with the closest points to the target speaker from the transferred points.
- *All*: The data points transferred are sampled at random from all the speakers. The set of inducing points $\mathbf{U}_m$ is initialised with a random subset of the transferred points.

and the three different speaker features presented in section 4.2.4:

- *Int*: Intelligibility features. For each speaker, $\mathbf{s}$ is defined as a single dimensional feature corresponding to the ineligibility measure provided by the UASpeech database.
- *APW*: Accuracy per word features. This is the same feature vector as the one defined in section 4.2.4. For each speaker, $\mathbf{s}$ is defined as the 36 dimensional vector corresponding to the ASR accuracy of each word in the command vocabulary.
- *IV*: I-vector features. For each speaker, $\mathbf{s}$ is defined as a 400 dimensional vector corresponding to the mean *i-vector* extracted from each utterance from that speaker. For more information on the *i-vector* extraction and characteristics, refer to Martínez et al. (2015).

The kernel over the speaker space k^s (equation 4.16), is defined as an RBF kernel (equation 4.17). This kernel is used both to compute the similarity between speakers in order to select data (section 4.2.2), and to weight the data from each source speaker (section 4.2.3). k^s has variance $\sigma_k^2 = 1$ and the lengthscale l_k varies depending on the features. For *Int* features $l_k = 0.5$, for I-Vector (IV) features $l_k = 8.0$ and for *APW* features $l_k = 4$.

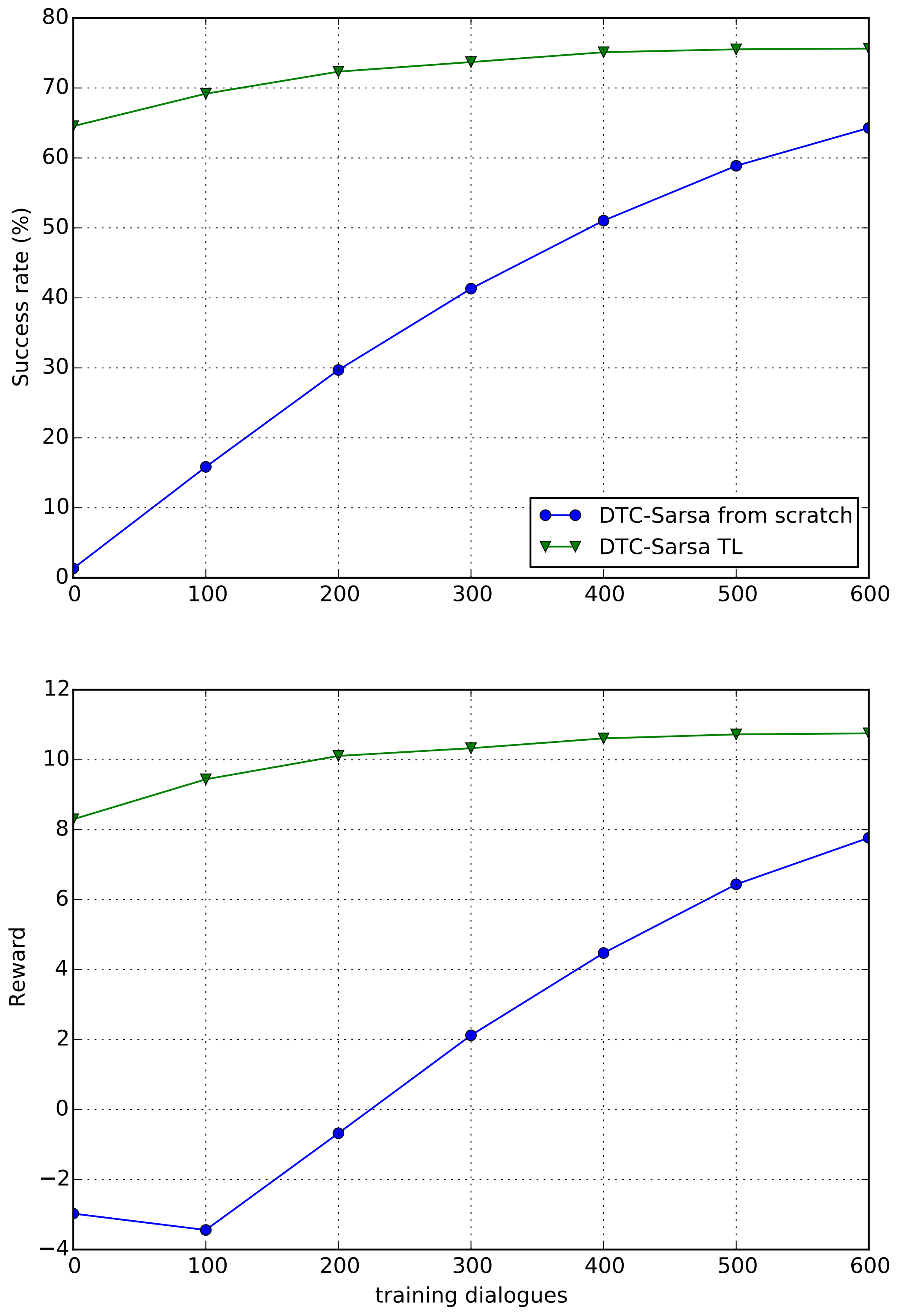

Figure 4.3: *Comparison of the number of dialogues needed to learn a DTC-Sarsa policy from scratch (blue) and with transfer learning initialization (green). The x axis represents the number of training dialogues.*

In the following experiments, the data to initialise each policy is transferred from a pool of 4200 points[1] corresponding to 300 points from each speaker in the UASpeech database (table 3.1) except for the speaker being tested. The size of the inducing set $\mathbf{U}_m$ is 500 and the maximum size of the TD points set $\mathbf{Z}_t$ is 1000 (or 2000 in some experiments). For online adaptation, whenever a new data point is observed from the target speaker, this point is added to the set of inducing points $\mathbf{U}_m$, and the first point of the set $\mathbf{U}_m$ is discarded from the inducing set (which, due to the ordering done by data selection, corresponds to the least similar source point or to the oldest target point). Whenever a new data point is observed and the size of the set of temporal difference points $|\mathbf{Z}_t| = 1000$, the first point of this set is discarded.

Experimental results

Figure 4.3 shows the effect of using data from source speakers to transfer knowledge to a policy model personalised to a target speaker. It compares the mean results of the six test speakers for a *DTC-Sarsa* policy model trained from scratch (*DTC-Sarsa from scratch*) and initialised with data from the other 14 speakers (*DTC-Sarsa TL*). The initialised policy uses the model *DTC* to transfer the data and does not use any speaker feature to weight or select the data. The data is transferred with the *All* selection method, selecting 1000 points at random from all the source speakers. As can be seen, transferring the data from source speakers gives a *Jump-start* improvement (performance when no target data has been seen) of more than 60% in success and more than 10 in reward. It can be also observed that, after initialising it with data from source speakers, the policy needs fewer dialogues (around 300) to converge to a near optimal performance. In contrast, when training from scratch, after 600 dialogues the policy is still far from converging.

To analyse the performance of the different models presented in section 4.2.1, figure 4.4 compares the three different policy models proposed (*DTC*, *Prior* and *Hybrid*), initialised with different amounts of source data points: 1000 (1K) and 2000 (2K). No speaker features are used to weight the data and the data selection method *All* is followed, selecting the data to transfer at random from the pool of source points. Analysing the results, it can be seen that for the three models, transferring more data points increases their performance when the amount of target speaker dialogues is small, at the cost of increasing the model complexity. As the amount of target speaker dialogues increases, however, the performance improvement obtained by transferring more source points decreases. When comparing the three different models, several observations can be made. When points from the

[1]Each pool is different for each of the different seeds used to initialise the experiments.

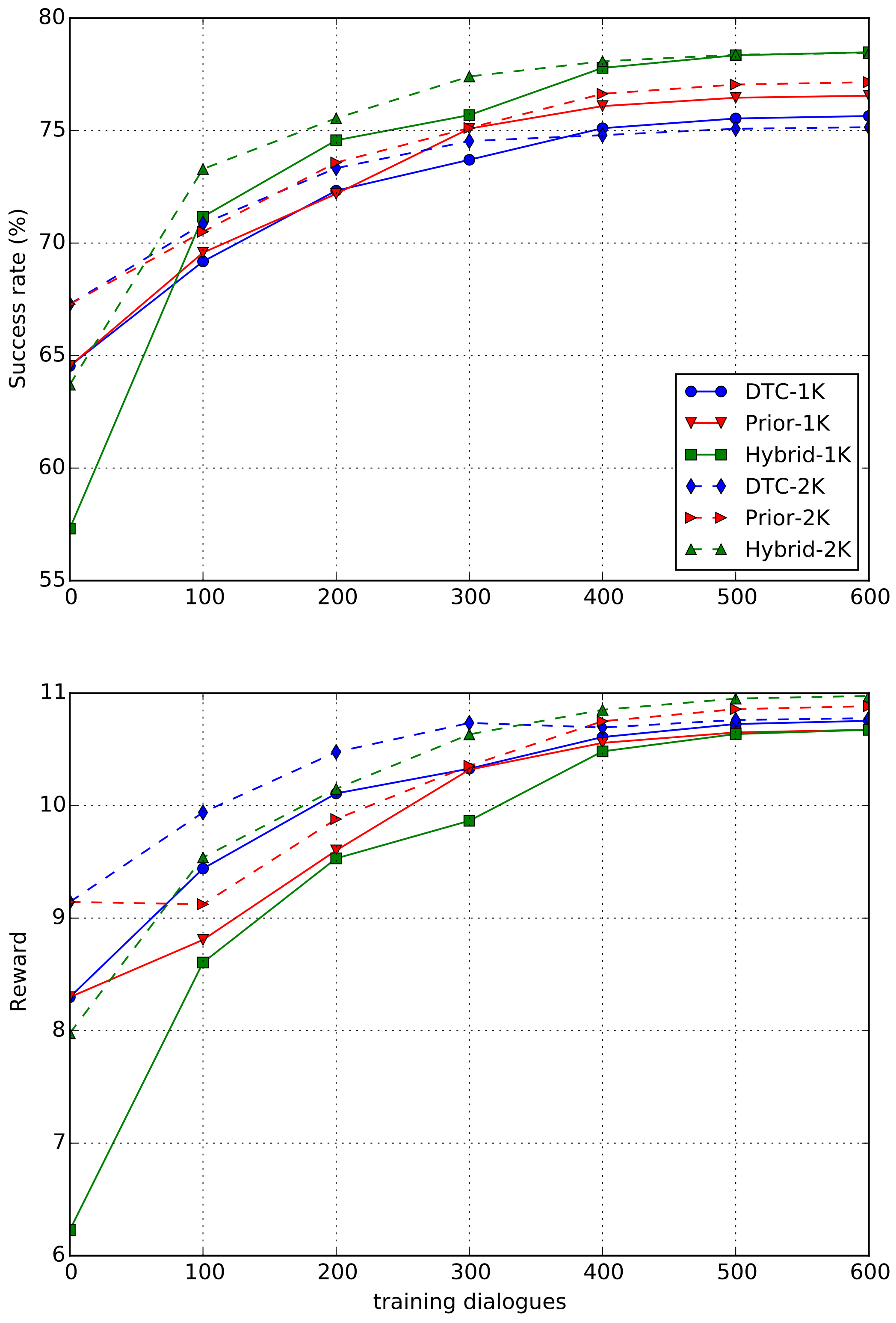

Figure 4.4: *Performance of different transfer learning models initialized with different amounts of source speaker data. The x axis represents the number of training dialogues and the dashed line that 2000 dialogues are transferred instead of 1000.*

target speaker have not been observed, *DTC* and *Prior* have the same performance (in this case, they are the same model indeed), but the performance of the *Hybrid* model is significantly bellow the two others. As data from the target speaker is included in the training set, the performance of the *Prior* model improves very slowly compared with the other two models, being quickly outperformed by the *Hybrid* model. The *DTC* model, has a much faster performance improvement than *Prior* as the number of observed target dialogues increases, but at the cost of a higher computational cost. Actually, the *Hybrid* model has the quickest performance improvement as function of the target speakers, quickly outperforming the other models in success rate (after 100 target dialogues) and getting a very similar reward with a large amount of target dialogues (after 400).

To compare the different approaches to compute the speaker similarities for data selection and weighting presented in section 4.2.4, figure 4.5 shows the performance of the DTC model with 1000 transfer points (named *DTC-1K* in the previous figure) using the different speaker features and data selection methods proposed. *Int* (blue) denotes that the model uses the intelligibility measure based features, *IV* (green) the *i-vector* features and *APW* (red) the ASR accuracy per word based features. *Clo* (continuous line) denotes that the 1000 transferred data points are selected from the most similar speakers and *All* (dashed line) denotes that the source points are transferred from all the speakers, sampling 1000 points at random from the pool of 4200 points. *None-All* is used as the baseline, where no speaker feature is used to weight the data and the data is transferred sampling at random from all the speakers.

Analysing the performance of the different speaker features evaluated, *IV* features are shown to outperform the baseline and the other two features, although *APW* features have a very similar performance. The better performance of *IV* features is maintained for any number of target speaker dialogues observed. The performance of *Int* features, however, is way below the other two metrics, performing even worse than not using any speaker feature to weight and select the data. If the two different data selection methods (*Clo* and *all*) are compared, it is surprising to see that the performance of *All* method is better than *Clo* for any of the features. This shows that the improvement obtained from using the proposed speaker features to weight the data is not obtained from using these features to select the data.

With the objective of analysing which is the best model/feature/selection combination, figure 4.6 plots the three transfer learning models (*DTC*, *Prior* and *Hybrid*) with the two best performing features (*IV* and *APW*) using the best performing data selection method, *All*. As it was seen in figure 4.4, the performance of the

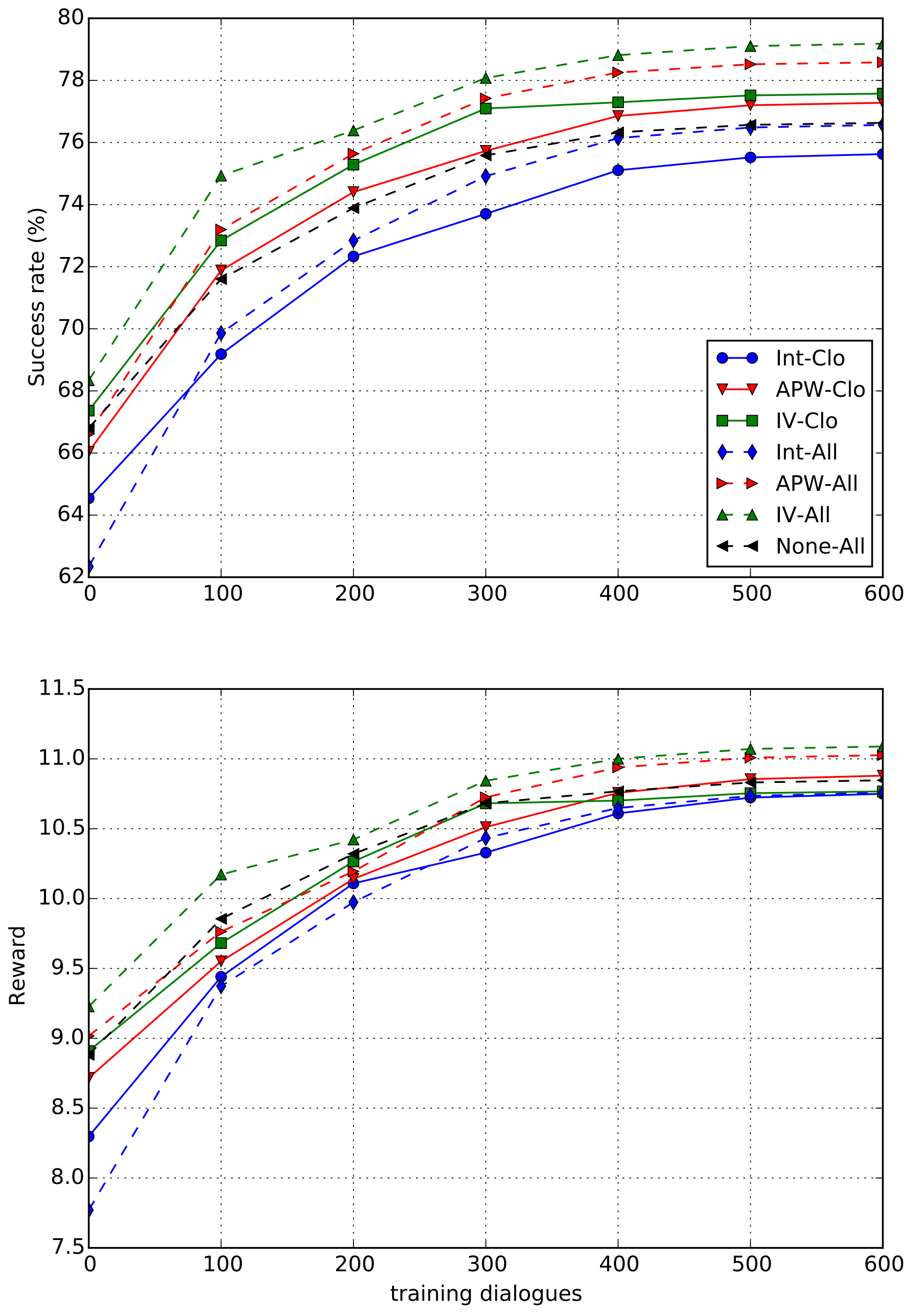

Figure 4.5: *Performance of different speaker features and different data selection methods. The x axis represents the number of training dialogues and the dashed line denotes that the data points are transfered from all the speakers instead of from the closest ones.*

Prior model is worse than the performance of the other two models. Comparing the *DTC* and *Hybrid* models, the former model has a much better performance when there is not target speaker dialogues, but the latter, after a few target speaker dialogues have been collected, quickly outperforms the other models in reward and success. Comparing the different features used, it can be seen that *IV* features still have the best performance, being generally slightly better than *APW* features for any amount of target speaker dialogues.

Experiment conclusions

These experiments have shown how initialising a dialogue policy personalised to a target speaker with data from other speakers improves the performance of the policy both in initial performance and in convergence speed to an optimal policy. In personalised VUIs, where collecting dialogue data from each target speaker is very costly, this method can give an initial "average" policy that can be quickly adapted to a near optimal policy as the user interacts with the system.

From the different transfer learning models evaluated, *Prior* showed worse performance than *DTC* and *Hybrid* in terms of the amount of dialogues needed to adapt the policy to the target speaker. Even if the computational complexity of *Prior* is smaller than *DTC*[1], the convergence speed decrease makes it a less suitable model for personalised VUIs. If a small amount of target speaker dialogues is available, the *Hybrid* model shows to be the best option for personalised VUIs, as it has a performance similar to *DTC* with a lower computational complexity.

When the different speaker features were compared, it was seen that *IV* features slightly outperformed *APW* features. As *APW* features use information about the ASR statistics (which is the input for the dialogue manager), it might be expected that they would outperform the rest, but in this case a purely acoustic based measure such as the one computed with *IV* features works better. The reason for this might be that these features are not totally correlated to the ASR performance, and there is hidden information inside *IV* features that help the model to find similarities between speakers better than the average ASR statistics. As these features are not strongly correlated, combining the two features could give further improvements. The performance of *Int* features, however, is way below the other two metrics and the baseline, suggesting that the information given by intelligibility assessments is a weak feature for source speaker selection. As this metric is computed in evaluations done by humans, it might be very noisy.

[1]Due to computing two parallel GPs, the complexity of *Prior* is $s * n^2 + t * m^2$ instead of the complexity of $(s+t) * (n+m)^2$ of the *DTC* model, where s and t is the number of source and target points respectively and n and m the number of source and target inducing points. The complexity of the *Hybrid* model is $(s/2) * (n/2)^2 + ((s/2) + t) * ((n/2) + m)^2$.

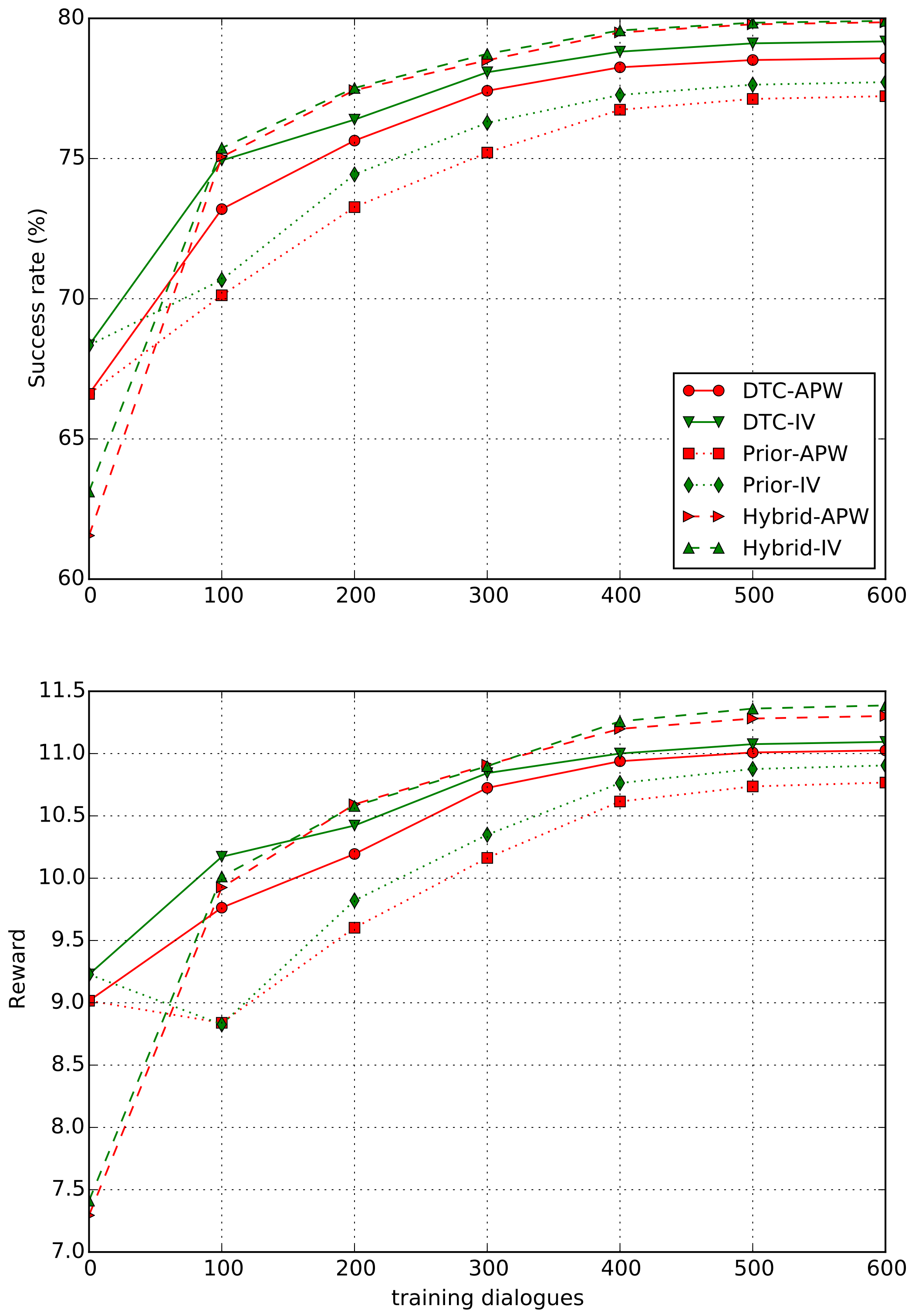

Figure 4.6: *Comparison of the best performing speaker features, APW (red) and IV (green), using different transfer learning models: DTC-Sarsa (continuous line), Prior (dotted) and Hybrid (dashed). The x axis represents the number of training dialogues.*

Evaluating the different data selection methods, the performance of *All* method was shown to be better than *Clo*. This suggests that transferring points from more speakers rather than from just the closest ones is a better strategy. This might seem counter intuitive, but if the belief-action points of each speaker are clustered in small regions of the belief-action space, transferring points from less speakers might lead to having parts of the belief-action space where information is not available. If the points are selected at random from more speakers, these points will be distributed more uniformly over the belief-action space. More sophisticated data selection methods that trade-off between filling the belief-action space while selecting the most similar points could increase the system performance.

Finally, when all the models were compared using the best features and data selection methods, the *DTC* model using *IV* features was shown to be the best when target speaker data is not available. After a few target speaker dialogues have been collected, however, the *Hybrid* model quickly outperformed the other models in reward and success. This suggests that, if a small amount of target dialogues is available, *Hybrid* combined with *IV* features is the best model to use in a personalised VUI, as it outperforms *DTC* model while having a lower computational cost.

4.5 Conclusions

Even if assuming that all the dialogues used to train a policy model are drawn from the same distribution is acceptable in several domains (Raux et al., 2005, Young et al., 2010), in cases such as personalised VUIs, this assumption may not be true. Two scenarios likely to occur in personalised VUIs where this assumption does not hold have been presented: variable environment (ASR) scenarios and scenarios where the policy is trained with data from several different speakers. In this chapter, a state-of-the-art, model-free, dialogue policy optimisation framework appropriate for these scenarios has been developed and evaluated. This framework is also appropriate for online policy adaptation to specific speakers.

One main requirement of this framework is the ability to arbitrarily select and discard data points. As online data selection and discarding is not tractable in the approximate GP-RL methods proposed in the literature (Engel et al., 2005, Gašić et al., 2013), two GP-RL methods that permit online data selection have been developed. The first one, named *GP-MC*, is based on modelling the RL problem with a *Monte-Carlo* approach, where the *Q*-function is approximated using the accumulated rewards instead of the immediate rewards. This method showed good performance when the amount of training dialogues is large, but the performance when the amount of training dialogues is small decreased. As fast learning rate and adaptability are important requirements for personalised policies, the usefulness of this method in hS like systems is reduced. Combining several policy models, however, has been shown to increase the dialogue performance (Gašić et al., 2015). Therefore, *GP-MC* can be useful combined with other policies that perform well with smaller amounts of training data.

The second proposed GP-RL method combined the definition of a kernel function in the *temporal-difference* space and the application of the *Deterministic Training Conditional* approximation method. Redefining the set of belief-action points into the TD space, allows to define a kernel function between TD points and rewrite the GP-RL equations with the same shape as GP regression models. This redefinition, allows to select and discard data points more easily (because the dependence between two consecutive belief-action points is well defined) and permits the application of well studied GP approximation methods such as DTC. This approach, named *DTC-Sarsa*, showed results comparable to *GP-Sarsa* with a computational cost tractable for online policy optimization, showing to be a very promising approach for personalised policy optimisation.

This chapter showed how initialising a personalised policy with data from other speakers can greatly boost the performance of the system when target speaker data is not available. In addition, once target speaker dialogues are collected through interaction, this initialisation can greatly speed up the convergence to an optimal policy. This is a desirable property for personalised dialogue policies, because a "speaker independent" policy can be trained, which has a performance good enough to be used by the target speaker. Later, this policy can be adapted online to a personalised optimal policy with only a few dialogues collected through interaction.

Another contribution of this chapter is a framework to weight the data collected from different probability distributions. To do so, each data point is extended with an *environment feature* (ASR environment or speaker environment) and the kernel in the belief-action space is extended to also compute covariances in the environment space. Including these extra features showed performance improvements in the two studied scenarios; variable ASR environment scenario and transfer learning between speakers scenario. The results, however, depended on the type of features used in each case, and some features even degraded the model performance. The two features that improved the results, *IV* and *APW*, are not strongly correlated, so combining them could lead to better results. Further investigation on more representative environment features, or the application of techniques that help to find the most representative features, could improve the usefulness of the data weighting approach. Different transfer learning models were also studied, and it was shown that combining a hybrid model with I-vector features outperforms any other approach after a few target speaker dialogues are collected.

Due to the redefinition of the set of belief-action points to the set of TD points, data points can be selected and discarded in an online fashion as more data points are collected from interaction with the user. In the scenario of transfer learning between speakers, different ways to select which points to transfer and discard were studied. It was shown how selecting data points from all the speakers outperformed selecting data points from the speakers closest in the speaker space. This is probably because there is more variability in the data points if they are selected from all the speakers, so the belief-action space is covered more uniformly. More sophisticated heuristics that trade-off between selecting the points closest in the speaker space and covering the most of the belief-action space could improve the performance.

In summary, this chapter has presented a tractable online policy optimization framework suitable for scenarios in where the training data comes from different distributions. This framework showed an increase in the dialogue performance in two different scenarios, but there are more potential scenarios in where this framework can be applied – e.g. dialogues in different noise conditions, dialogues with different accent speakers, etc.

Chapter 5

Personalised Dialogue State Tracking

In the research presented in this thesis up to this point, little attention has been paid to Dialogue State Tracking (DST). In the dialogue community, interest in DST has been growing since the first Dialogue State Tracking Challenge (DSTC) was held in 2013 (Williams et al., 2013) and has continued with the following DSTCs (Henderson et al., 2014a,b, Kim et al., 2016). In DM for VUIs personalised to dysarthric speakers, dialogue state tracking plays a crucial role in the outcome of the dialogue, as it is the module in charge of estimating the true user intention or goal from the noisy observations of the ASR or SLU. The dialogue state tracker is therefore the module that increases the robustness in challenging ASR environments. However, to the author's knowledge, no research has been conducted so far in personalisation or user adaptation of dialogue state tracking models to specific users or environments. In this chapter, two scenarios likely to appear in personalised VUIs are presented, showing that the use of features related to the environment can help the dialogue state tracking models to improve the generalisation to unseen speakers or dialogue states. These scenarios are described in the next two subsections.

Environment features for Dialogue State Tracking

In the previous chapter, the concept of personalisation of dialogue policy by extending the input features with *environment features* was introduced, giving evidence that it can improve the performance of dialogue policies trained with data from source speakers. However, the approach taken to introduce the environment features in the GP-RL model has some shortcomings. First of all, the correlations between points from different sources depend on a defined kernel with hand-tuned hyper-parameters. Even if research has been done in optimising the kernel hyper-parameters automatically (Chen et al., 2015), the method turned out to be unstable when employed in the hS environment. Second, as the kernel is factorised into kernels in different spaces, the model cannot represent correlations between different parts of the belief space and the environment space. Therefore, the inclusion of the environment features in these models can be summarised as computing the similarity between the different environment spaces of two points, and using this similarity as a weighting factor when computing the covariance between two belief-action points. Discriminative models can make a more efficient use of the environment features, because they can model the correlations between individual elements of the environment features with the rest of the input features (Bishop, 2006, Henderson et al., 2012).

Since the environment features are extracted before the belief state is estimated, it would seem more reasonable to use them during the DST step. Conveniently, discriminative models have recently been shown to have the best performance in DST tasks (Henderson et al., 2014c, Lee, 2013, Williams, 2014), precisely because of their capacity to use higher dimensional, possibly correlated, input features, by directly modelling the conditional probability of the dialogue state given the input features (Lee and Eskenazi, 2013). In section 5.2, a Recurrent Neural Network (RNN) based dialogue state tracker able to use extended environment features is presented and evaluated.

Dialogue state tracking generalisation to unseen states

There are other situations where DST techniques can be helpful in a hS environment. For example, as explained in section 2.1.1, when the ASR performance of a homeService system has improved until a convergence point, the researchers may opt for extending the range of devices that the system can control, extending the ASR vocabulary. This means that, for DST, the researchers will have access to data collected in the reduced domain, but would need to train a tracker that can understand new concepts. Another possible (and similar) hS scenario is one in which the user changes one of the devices (e.g. buys a new TV) and thus needs a different set of commands to control it. The third DSTC (Henderson et al., 2014b) investigated this same issue, evaluating state trackers in extended domains by including dialogue states in the test data not seen during the training. This challenge showed the difficulty for data-driven approaches to generalise to unseen states, as several machine learnt trackers were outperformed by the rule-based baseline. Effective DST generalisation to unseen dialogue states (e.g. changing the dialogue domain or extending it) remains an issue.

State trackers using slot-specific models, however, cannot handle dialogue states not seen in the training data. This is because the model is trained as a classifier in which the classes are the possible dialogue states. Hence, if a state is not seen in the training data, the model will never learn to classify it as the correct one. Therefore, researchers participating in the third DSTC had to develop state trackers able to generalise to unseen dialogue states, sometimes named *general dialogue state trackers* (Henderson et al., 2014d). One of the most promising approaches to general DST is to use state trackers that track each state independently by using general value-specific features (Henderson et al., 2014c, Mrkšić et al., 2015).

However, dialogue states are by definition in a discrete space where similarities cannot be computed. Thus, a general state tracker has to include a general value-tracking model that learns the average tracking statistics of all dialogue states. This strategy assumes that different dialogue states have the same state tracking behaviour, but such assumption is rarely true. For example, two values, whose associated concepts have different ASR accuracy, have different state tracking performance. If a general feature which can be used to compute similarities between dialogue states could be defined, the state tracking generalisation to unseen states could be improved. The unseen dialogue states could be tracked using statistics learnt from the most similar states seen in the training data instead of using the average statistics of all states. In section 5.3, a general dialogue state tracker able to better generalise to unseen dialogue states by using ASR and phone related features is presented and evaluated.

Chapter overview

In the rest of the chapter, techniques to improve the generalisation to unseen speakers and to unseen dialogue states are presented, scenarios that are likely to be found in hS-style systems. To properly evaluate these two scenarios, however, it is more convenient to study each of them independently. In addition, a dialogue corpus of dialogues occurring in these scenarios is needed. The next section describes the generation and characteristics of the dialogue corpus used to evaluate the DST experiments performed in this chapter. Then, section 5.2 investigates techniques aiming to improve generalisation to data from unseen speakers in an LSTM-based state tracker. The DST input features are extended with speaker specific features (*i-vectors* and ASR-related) to help the model to find similarities between the target and the source speakers. *dropout* regularization (Srivastava et al., 2014) is also used, showing that it helps not only to generalize to unseen speakers, but also to increase the performance improvement of the tracker when using the augmented input features. In a further analysis, it is shown that the effect of these generalization techniques increases when a small amount of target speaker data is available. Section 5.3 develops a method to use ASR and phone-related general features to improve the generalisation of a RNN based dialogue state tracker to unseen states. Several ASR and phone-related features are proposed and evaluated as well as different approaches to encode variable length phone sequences into fixed length vectors. Finally, section 5.4 draws a conclusion for the chapter.

5.1 Dialogue state tracking corpus for hS environment

One of the main problems in dialogue management research is the lack of annotated dialogue corpora and the difficulty of using data from one domain to train a system in a different domain. The corpora released for the first three DSTCs aimed to mitigate this problem (Henderson et al., 2014a,b, Williams et al., 2013). However, this data has been collected in a scenario where many different speakers interact a few times each, thus making adaptation to specific speakers infeasible. Furthermore, there is no acoustic data available; hence, features extracted from the acoustics, such as i-vectors or phone posterior probabilities, cannot be used. For these reasons, the dialogue corpus used in the experiments in this chapter has been generated with simulated users interacting with a rule-based dialogue manager. The simulated users interact with ASR systems adapted with two different amounts of speaker specific words: 0 (non adapted) and 300 (fully adapted).

To evaluate techniques to generalise to unseen speakers, data collected from several dysarthric speakers is needed, whereby each speaker interacts for a large number of dialogues. To do so, a set of 15 SUs with dysarthria (the same SUs described in section 3.2.3) is used to interact with a rule-based dialogue manager and the dialogues generated are used as the corpus to train and evaluate the DST models. The interaction is performed in the simulated hS environment described in section 3.2.3, using the slot-based dialogue state representation approach described in section 3.2.2. However, stochastic factors, such as the SUs and exploratory policies, influence the corpus generation. To reduce the effects introduced by these random components, three different corpora have been generated by initialising the random components of the users and the policy with different seeds. Then, 1200 dialogues are collected for each speaker-ASR pair and for each seed and the results presented in this chapter are the mean results of the tracking evaluation on the three corpora.

To evaluate techniques that improve the generalisation to unseen states, dialogue data collected in different or extended domains is needed. For the experiments performed in this chapter, data collected in two different domains (corresponding to different hS systems, systems that use a different set of commands to be controlled) has been generated. Then the data from one domain is used to train the DST models and the data from the other is used to evaluate the models. This approach, instead of an extended domain approach as in DSTC 3, is taken for one main reason: the main interest is to evaluate the performance of a tracker when tracking unseen states. If both seen and unseen states are included in the test data, it would not be clear which part of the results correspond to the tracking of the seen states and which of them correspond to the tracking of the unseen ones. To generate the corpus, two hS systems are simulated, each controlled with a different vocabulary of 36 commands. To do so, 72 commands selected at random from the set of 155 most frequent words in the UASpeech database (Kim et al., 2008) are split into 2 groups, named *domain A* and *domain B*. Then, 1000 dialogues are collected for each speaker-ASR pair in each domain[1] and used as corpus for the state tracking experiments in section 5.3. To ensure that the methods are independent of the set of commands selected, three different 72 word vocabularies are randomly selected and the results presented in the following section show the mean results for the three vocabularies.

5.1.1 Slot-based dialogue state representation for homeService

In contrast to the dialogue managers presented in the previous chapters, DST techniques have to be evaluated using more complex dialogue state representations. In this chapter, the slot-based approach presented in section 3.2.2 is used. The dialogue state of the system is factorized into three slots, with the values of the first slot representing the devices to control ("TV", "light", "bluray", etc. blue commands in figure 3.6), the second slot its functionalities ("channel", "volume", etc. green commands) and the third slot the actions that these functionalities can perform ("up", "two", "off", etc. red commands). The slots have 4, 17 and 15 values respectively and the combination of the values of the three slots compose the joint goal (e.g. "TV-channel-five", "bluray-volume-up"). In the case of user goals composed by two values, the third slot takes a special value, "none" (e.g. the goal "light-on", will be "light-on-none"). The set of valid[2] joint goals $\mathcal{G}$ has a cardinality of 63 and the belief state $P(g)$ for each joint goal g[3] is obtained by multiplying

[1]200 extra dialogues are collected in *domain B* for the set of experiments which includes a small set of in-domain data.

[2]Take into account that many combination of slot values will not be valid, e.g. light-channel-on.

[3]The joint goals g are used instead of the dialogue states s to make clear that it does not refer to the value of a slot. Anyway, a policy could take as input the joint goal distribution or the factorised

the slot probabilities of each of the individual slot values and normalising:

$$P(g) = \frac{P_{s1}(g_1)P_{s2}(g_2)P_{s3}(g_3)}{\sum_{g' \in \mathcal{G}} P_{s1}(g'_1)P_{s2}(g'_2)P_{s3}(g'_3)} \tag{5.1}$$

where $P_{sx}(g_x)$ is the probability of the value g_x in slot s_x and $g = (g_1, g_2, g_3)$.

The slot-based dialogue state tracking corpus is labelled following the annotation *scheme B* defined in the DST challenges (see section 2.7.3).

5.1.2 Simulated dysarthric users

To generate the dialogue corpora, a set of simulated users has been used. These SUs are composed by a *behaviour simulator* and an *ASR simulator*, following the same approach as the ones described in section 3.2.3. The dialogues are generated for the 15 dysarthric speakers in two ASR adaptation regimes (0 and 300), making a total of 30 speaker-ASR pairs. As in the previous chapter, only the DST performance for low and mid intelligibility speakers is evaluated. However, in order to interact in a mixed initiative environment and to generate the necessary acoustic features, some modifications of the SUs are needed.

The set of rules to control *behaviour simulator* is modified to interact in a mixed initiative slot-based environment. The set of rules is the following:

- The user will speak the commands corresponding to the user goal in any random order (e.g. it can say channel-five-TV instead of TV-channel-five).
- If the system asks for the goal, the user will say a command corresponding to the value of a slot that has not been uttered before, or any command of the goal if all the values have been said.
- If the system asks for the value of a specific slot, the user will say the value of that slot.
- If the system asks a confirmation question, the user will answer with yes or no.

To simulate different levels of expertise by the user, the simulated user can "confuse" the command it has to say: in each turn and depending on the user expertise, there is a probability of saying a different command or of providing a value for a different slot than the one requested. Three different expertise levels are used to generate the corpus to increase its variability. Refer to appendix B for a deeper explanation of the behaviour simulation model.

distributions for each slot. In the former case, $\mathcal{G} = \mathcal{S}$ and $P(g) = b(s)$.

In order to be able to generate acoustic features better correlated with the N-best lists, the *ASR simulator* follows the sampled ASR approach explained in section 3.2.3, in which the ASR N-best output is sampled from the ASR outputs obtained in the experiments in section 3.1.2. For each command generated by the behaviour simulator, an N-best list output from one of the recordings of that command for that speaker-ASR pair is sampled at random. To increase the variability of the data generated, the time scale of each recording is modified to 10% and 20% slower and 10% and 20% faster, generating more ASR outputs to sample from. To generate phone posterior features (used in section 5.3) the approach described in Christensen et al. (2013a) is used, without the principal component analysis dimensionality reduction.

Rule-based state tracker

To generate the corpus, the simulated users described in the previous section interact with a rule-based dialogue manager. The dialogue state tracker of this dialogue manager is one of the state trackers used as baseline (Wang and Lemon, 2013) in the second DSTC. This tracker follows simple rules to compute the belief state as the accumulated evidence seen in the SLU output (or ASR output in this case) during all the dialogue history. This tracker performed competitively in the DSTCs, proving the difficulty for data driven trackers when the training and test data are mismatched. The state tracking accuracy of this tracker is also used as the baseline in the following experiments.

Rule-based dialogue policy

The dialogue policy employed to generate the corpus is an improved version of the rule-based policy used as baseline in the experiments in section 3.4. This policy is modified to include mixed initiative interaction strategies. It follows simple rules to decide which action to take in each turn:

- For each slot, if the maximum belief of that slot is below a threshold, t_a, the system will ask for that slot's value.
- If the belief is above the threshold t_a but bellow a second one, t_b, the system will ask a confirmation question for the value.
- If the maximum beliefs of all slots are above the threshold t_b, the system will take the action corresponding to the top ranked joint goal (the joint goal with the highest probability).

The values for the thresholds t_a and t_b are optimized to each speaker-ASR pair by grid search to maximize the dialogue reward. In addition, the policy implements a stochastic behaviour to induce variability among the collected data; choosing an action at random with probability p_c and requesting the values of the slots in a random order. To increase the variability of the corpus, the dialogues are generated using two different values for p_c: 0 and 0.2.

5.2 DST feature extension for improved generalisation to unseen speakers

Recalling section 2.5, the component in charge of inferring the dialogue state in each turn is called the *dialogue state tracker*. This module takes the dialogue history as input (the collection of ASR-SLU observations, machine actions, etc. up to the current turn) and uses it to estimate the distribution over the dialogue states – the belief state.

Historically, machine learning approaches to DST used generative models, which need to model all the correlations in the input features (Thomson and Young, 2010, Williams and Young, 2007). This forced the generative models to make many conditional independence assumptions in order to maintain tractability. In addition, to keep the input features as lower dimensional and uncorrelated as possible, only the *dialogue features* (the SLU output plus last system action) were used as input features

The resulting SDSs used a typical "pipeline" architecture with very defined data flow (shown in figure 5.1 as the continuous line). The data flow in this architecture started with the user utterance (acoustic signal), being transformed to a string of words by the ASR and to a set of concepts by the SLU, then being feed to the state tracker of the dialogue manager and so on. In this architecture, each module reduces the data dimensionality, trying to keep only the key information needed to infer the dialogue state. However, some useful information could be lost in each step.

In the recently held DSTCs (Henderson et al., 2014a,b, Williams et al., 2013), it was shown how discriminative models outperform generative models in DST, because of their capability to incorporate a rich set of features without worrying about their dependencies on one another. Most models used very high dimensional input features generated from the whole dialogue history (Lee, 2013, Williams, 2014) while others even extracted the features directly from the ASR output (Henderson et al., 2014d). Discriminative models open up the possibility to extend the tracker's input features with features extracted in previous modules of the dialogue system.

In chapter 4, environment features such as i-vectors and ASR performance related features were shown to improve the generalisation of a dialogue policy to unseen dysarthric speakers. However, the ability of discriminative models to handle high dimensional input features suggests that the information obtained from these environment features could be more efficiently processed in the DST step instead of in the PO step. For example, in personalised dialogue management, the dialogue features can be extended with the user specific features described in section 4.2.4 – i-vectors and ASR performance related features (*APW*). These features give information that represent a certain type of speaker behaviour, which allows the state tracker to relate it to the behaviour observed on "similar" source speakers when inferring the dialogue state. Using the environment features proposed in chapter 4 with discriminative model based dialogue state trackers can have several advantages:

- The deviations in the estimation of the belief state in each turn due to the mismatch between the training and testing data can be "corrected": the state tracker can use the information included in the environment features to compensate the "noise" introduced by differences between the speakers or environments. Therefore, the policy optimization step will work with an "environment corrected" input, meaning that it can be considered that all the input data for the policy (the belief state) comes from the same distribution, thus the same policy can be trained with data from several (possibly very different) speakers or from different ASRs.

- Discriminative models can model the correlations between the individual elements of the environment features and the DST input features. For example, a discriminative model can find that a specific dimension of an *i-vector* is correlated with the accuracy of a specific command in the N-best output of the SLU.

- Discriminative models (e.g. RNNs) can learn their weights automatically with gradient descent methods. This means that fewer model parameters have to be hand-tuned and that the model has more flexibility to learn the true distribution of the data.

Therefore, to improve the DST generalisation to unseen speakers, the usual dialogue data flow can be modified to include features extracted directly from the acoustic signal and from the ASR (shown in figure 5.1 as the dashed line), such as the the speaker features explained in section 4.2.4. More formally, in each turn t,

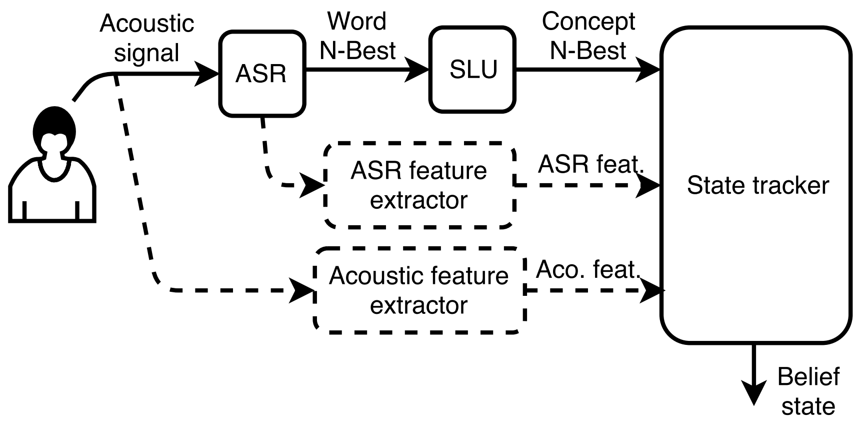

Figure 5.1: *Typical dialogue data flow (continuous line) and proposed extended dialogue data flow (dashed line).*

the dialogue features $\mathbf{o}_t$ can be extended by concatenating them with the speaker features $\mathbf{o}_s$. Following this approach, the input of a (sequential) state tracker in each turn will be the concatenation of these two features, $\mathbf{o}_t \oplus \mathbf{o}_s$.

5.2.1 Experimental set-up

This section presents a framework to evaluate if including speaker features in the input of the state tracker can improve the DST generalization to speakers not seen in the training data. To do so, the methods proposed in section 5.2 are tested on a set of slot-specific LSTM-RNN (Hochreiter and Schmidhuber, 1997) based state trackers. These trackers follow the slot-based dialogue state representation described in section 5.1.1. To evaluate the setting up of a system where dialogue data from the target speaker is not available (the same TL set-up presented in the experiments in section 4.4.4), the tracker for each speaker is trained on data from the remaining 14 source speakers. A second set of experiments tries to evaluate the performance of the trackers as target speaker data becomes available by including increasing amounts of target speaker dialogues the training data. The performance of the state trackers is evaluated on the six SUs corresponding to the low and mid intelligibility speakers in the UASpeech database, the test speakers.

RNN-LSTM model parameters

To train the state tracking models, 1200 dialogues are used for each source speaker (with a 0.9-0.1 train-validation split) and 1200 target speaker dialogues are used for testing. In the second set of experiments, increasing amounts of target speaker dialogues are used to train the model, including 200 extra target dialogues in each step[1]. The RNN-LSTM models are trained for 100 iterations with stochastic gradient descent with a learning rate of 0.001.

As the data gathered from the source speakers used for training will follow a different distribution than the target speaker data, the tracker might overfit to the source data. To address this issue, strong regularisation techniques are needed. Since model combination was shown to be important in the performance of the best performing trackers in the three DST challenges (Henderson et al., 2013, 2014b, Williams, 2014), the outputs of the five models corresponding to the five iterations performing best in the validation set are combined to get the slot output distribution. However, model combination might not be enough in scenarios with very big data mismatch. Dropout regularization (Srivastava et al., 2014) has been proven to be a powerful regularization technique for artificial neural networks, greatly improving the performance in several machine learning tasks (Krizhevsky et al., 2012, Zaremba et al., 2014). Dropout randomly "deactivates" a percentage of neurons in each layer at every training iteration, forcing neurons to learn activation functions independent of other neurons. It can be considered a very extreme instance of model combination, where each training iteration is done with a different model. However, RNNs and especially LSTMs are difficult to train, so dropout can make it more complicated to learn long term dependencies (Bayer et al., 2013). To avoid this issue, dropout is only applied in the non-recurrent connections between layers as proposed by Zaremba et al. (2014). The dropout rate is set to 0.2 in the input connections and 0.5 in the remaining non-recurrent connections.

The state trackers are modelled as slot-specific trackers; for each slot an RNN-LSTM classifier is defined, in which the classes correspond to the possible values of the slot. The topology of the network is shown in figure 5.2. The input in each turn for each slot is composed by the dialogue features $\mathbf{o}_t$ (ASR N-best output plus the machine action) concatenated with the speaker features $\mathbf{o}_s$ (if any). The input is fed into a linear projection layer that in turn feeds into a recurrent LSTM layer. The output of the LSTM layer is the input to a *softmax* layer with a size equal to the number of slot values. Two different linear-LSTM layer sizes have been tested: 25-75 (*SML*) and 75-150 (*LRG*)[2]. Each model is evaluated with and without using dropout in training, with dropout rates of 20% in the input connections and 50% in the rest. This defines a total of four RNN-LSTM based trackers evaluated in section 5.2.3, named *SML*, *SML-DO*, *LRG* and *LRG-DO* respectively.

[1]The target speaker dialogues used for training are independent of the target speaker dialogues used for testing.

[2]The reason to compare LSTMs with different sizes is because dropout reduces the effective size of the network (Srivastava et al., 2014), thus optimal network sizes might vary depending on the dropout rate. Several network sizes have been tested and the two with better performance with and without using dropout are presented.

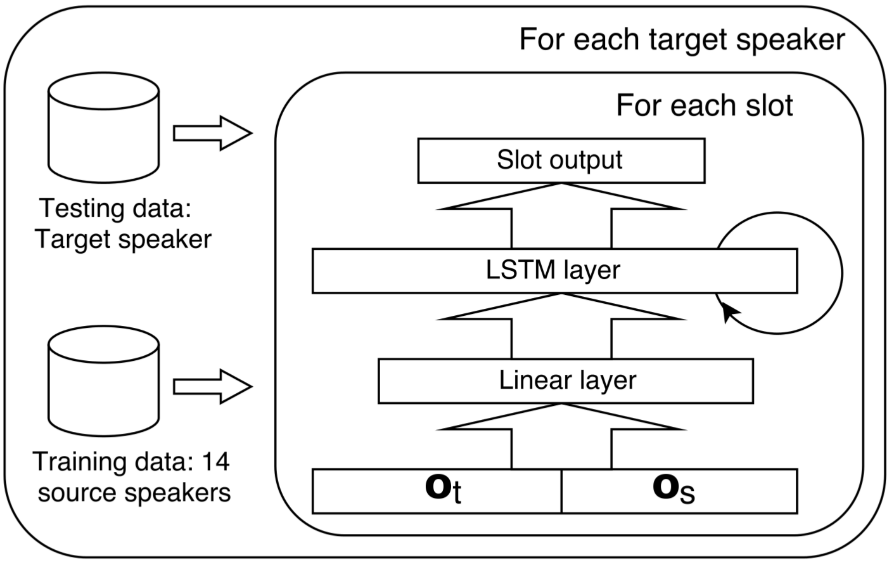

Figure 5.2: *Topology of the LSTM-based tracker.* $\mathbf{o}_t$ *and* $\mathbf{o}_s$ *represent the dialogue features and the speaker features, respectively.*

5.2.2 Extended input features

The standard input features of the tracker in each turn $\mathbf{o}_t$ are the dialogue features, i.e. the N-best list of commands outputted by the ASR plus the machine action in turn t. In section 4.4.4, it was shown that *i-vectors* and ASR performance related features could improve the performance of a policy trained with data from other speakers. Therefore, these features were used as speaker features in the following experiments. The DST models are evaluated concatenating the dialogue features $\mathbf{o}_t$ with the following speaker features $\mathbf{o}_s$:

- *IV*: Martínez et al. (2013) showed that *i-vectors* (Dehak et al., 2011) can be used to predict the intelligibility of a dysarthric speaker. Therefore, they are a potentially useful feature to relate similar speakers. For each speaker s, $\mathbf{o}_s$ is defined as a 50 dimensional vector corresponding to the mean *i-vector* extracted from each (test) utterance from that speaker in the UASpeech database. For more information on the *i-vector* extraction, refer to Martínez et al. (2015).

- *APW*: As explained in section 4.2.4, the statistics of the ASR can be used as speaker features. For the following experiments, the accuracy per word (command) is used, defining $\mathbf{o}_s$ as a 36 dimensional vector in which each element is the ASR accuracy for each of the 36 commands.

- *IV+APW*: As is was discussed in section 4.5, *IV* and *APW* features are not

Tracker		Speaker features *no feat.*	*IV*	*APW*	*IV+APW*
Baseline	acc.	64.8%$^{-}$	-	-	-
	L2	0.66	-	-	-
SML	acc.	66.9%	65.2%$^{-}$	66.6%	67.1%
	L2	0.482	0.501	0.484	0.483
SML-DO	acc.	67.3%	68.7%$^{+}$	70.1%$^{+}$	70.6%$^{+}$
	L2	0.451	0.427	0.418	0.408
LRG	acc.	66.1%	66.2%	66.5%	68.6%$^{+}$
	L2	0.497	0.505	0.489	0.464
LRG-DO	acc.	67.4%	69.7%$^{+}$	69.7%$^{+}$	70.0%$^{+}$
	L2	0.459	0.427	0.424	0.417

Table 5.1: *State tracking accuracy and L2 results for the different trackers using different speaker features. SML (25-75) and LRG (75-150) is the size of the layers and DO indicates that dropout is used. IV are i-vectors and APW accuracy per word features. +/- indicates statistically significantly better/worse than the SML tracker ($p < 0.01$), computed with a two-proportion z-test (Lehmann and Romano, 2006).*

strongly correlated. Therefore, defining $\mathbf{o}_s$ as the concatenation of *APW* and *IV* can further improve the performance of DST.

5.2.3 Experimental results

To show the performance of including speaker features when there is no target speaker data available, table 5.1 shows the state tracking accuracy and *L2* mean results for the six test speakers, when the trackers are trained with data from the source speakers only. The dialogue state tracker used as baseline (first row) is the best baseline state tracker used in the second and third DSTCs (this is the same rule-based state tracker used to collect the corpus in section 5.1, refer to Wang and Lemon (2013) for more details). The remaining four rows present the results for the four RNN-LSTM based trackers presented in section 5.2.1 – *SML*, *SML-DO*, *LRG* and *LRG-DO*. The columns denote the speaker features $\mathbf{o}_s$ that are concatenated to the dialogue features $\mathbf{o}_t$ in the tracker's input (the features presented in section 5.2.2), while *no feat.* denotes that only the dialogue features are used as input.

First of all, note that the accuracy and *L2* results are very strongly correlated in all the models (for *L2*, a lower value denotes better performance). Therefore, the analysis done in this section will focus only on the accuracy results, but it is applicable to the *L2* results as well. When the RNN-LSTM based trackers do not use speaker features or dropout (*SML-no feat.* and *LRG-no feat.*), the absolute accuracy increase with respect to the baseline is only 2.1% and 1.3%, respectively. When dropout regularisation is included (*SML-DO-no feat.* and *LRG-DO-no feat.*),

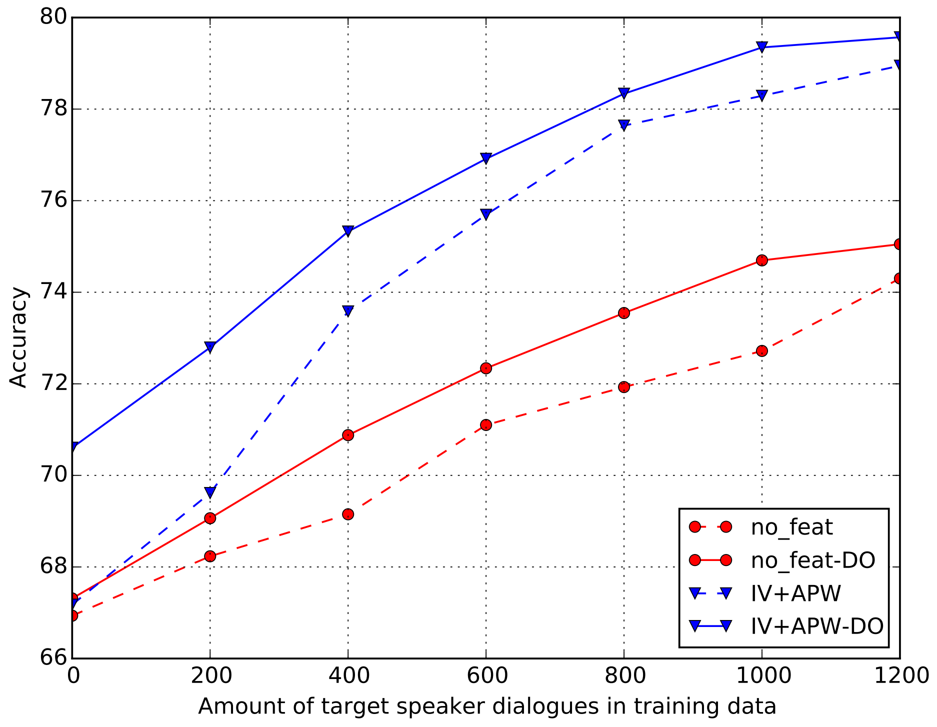

Figure 5.3: *Accuracy for SML tracker, using different amounts of target speaker dialogues in the training data. DO indicates that dropout regularization is applied and IV+APW indicate that the concatenation of IV and APW features is used.*

the accuracy improvement increases to 2.5% and 2.6%, respectively. When *APW* and *IV* features are included independently and dropout is not used, the results vary between the different models. For *SML*, the performance with respect to not using any speaker feature is degraded, while for *LRG* the performance slightly increases. When the concatenation of both features is used, however, the accuracy of *SML-IV+APW* slightly increases and the accuracy of *SML-IV+APW* increases by more than 2.5%. Including dropout regularization improves the accuracy of all the models, but the performance increase is considerably larger when *APW* or *IV* features are used, with improvements between 1.5% and 3% absolute. Using the concatenation of both features plus dropout (*SML-DO-IV+APW* and *LRG-DO-IV+APW*) results in the largest accuracy improvement with respect to the baseline, 5.7% and 5.1% respectively. These models also give the best accuracy improvement with respect to the "baseline" RNN-LSTM trackers (*SML-no feat.* and *LRG-no feat.*), 3.6% and 2.6% respectively.

To evaluate the performance of the trackers in an online adaptation scenario, figure 5.3 shows the accuracy of the *SML-no feat.*, *SML-DO-no feat.*, *SML-IV+APW* and *SML-DO-IV+APW* trackers when different amounts of user specific dialogues are included in the training set. The results show that the accuracy improvement obtained from including speaker features increases as more target speaker dialogues are included in the training set, obtaining more than 4% absolute improvement compared with not using speaker features for any amount above 400 dialogues. When a small number of target speaker dialogues is included in the training set, however, the gain obtained from the combination of speaker specific features and dropout regularization (*SML-DO-IV+APW*) is significantly higher than any of these approaches alone (e.g. 3% with 200 dialogues). As more target speaker data is included in the training set, the gain obtained from the *IV+APW* features increases with respect to the gain obtained from dropout, even if *SML-DO-IV+APW* still performs around 1% better.

Experiment conclusions

These experiments show how including speaker specific features as input for the state tracker can improve the performance in conditions of mismatch between the training and the test data. It can be seen that the accuracy of the baseline tracker is only around 2% below the performance of all the RNN-LSTM trackers when speaker specific features are not used, even if dropout regularisation is applied. This shows the difficulties faced by machine learnt models in mismatched train-test data conditions. Including *IV* or *APW* features independently without using dropout, degraded the performance in some cases and slightly increased it in others. When the results were analysed speaker by speaker, it was observed that, using *APW* and *IV* features independently degraded the performance for some speakers and improved it for others. This suggests that for some speakers the best speaker similarity measure is computed using *APW* features and for others, *IV* features work better. By combining both, the RNN-LSTM tracker is able to learn which features work best for a certain type of speaker.

The best results were obtained when dropout regularisation was used in combination with the different speaker features. The cause of this is probably that extending the input increases the chance of the networks to overfit, because neurons learn co-adaptations to detect fine patterns that only occur in the training data. By using dropout, these co-adaptations cannot be learnt because the presence of any particular input is unreliable. Therefore, when dropout is used, the trackers are forced to learn the *key* information appearing in the speaker features while the “noise” is discarded.

When the results were analysed including increasing amounts of target speaker data in the training set, it was shown how the improvement given by applying dropout to the models using speaker features decreased as more target dialogues were included in the training data. The reason for this is that dropout is more useful when the mismatch between training and test data is greater. As more target dialogues are included in the training data, the model is able to use the speaker features to relate the state tracking statistics of different speakers without the need of a strong regulariser.

5.3 DST feature extension for improved generalisation to unseen states

When a dialogue state tracker is modelled as a slot-specific tracker (e.g. the trackers in the experiments in section 5.2), the output classes of the classifier of each slot define the values that the slot can take. However, if a specific value for a slot does not appear in the training data or appears very few times, the model will not be able to learn to track it properly. Therefore, data driven state trackers with slot-specific models are not able to generalise to unseen dialogue states[1], because they learn the specific statistics of each slot and value. If a value is never seen in the training data, the model will learn that the statistics of that value say that it is never the true class. To handle this issue, general state trackers were proposed (Henderson et al., 2014d, Mrkšić et al., 2015), which track each value independently using general value-specific features (see section 2.5.3 for a more detailed review). These trackers define a set of binary classifiers or *value filters* for each slot-value pair, with all the value filters sharing the same parameters. The only difference between the value filters is the input, which is composed by value specific features $\mathbf{o}_v^t$ (e.g. the confidence score of the concept associated to that value in the SLU output). To compute the output distribution of each slot, the outputs of each independent value filter for that slot g_v^t are concatenated and normalised using a softmax function (see figure 2.6).

5.3.1 Similarities between different dialogue states

The main problem faced by general state trackers is that dialogue states are by definition in a discrete space where similarities cannot be computed. Therefore, a general state tracker has to include a general value-tracking model learnt from the average statistics of all the dialogue states. This strategy assumes that different dialogue states have the same state tracking behaviour, but such assumption is

[1]In a slot-based dialogue system, the dialogue states are defined as the set of possible value combinations for each slot. However, in this section the term *dialogue states* will be used to refer to the set of *slot-value* pairs and *joint dialogue states* to the actual dialogue states.

rarely true. For example, in a SDS for tourist information, it seems reasonable to assume that the state tracking statistics of the state "search-restaurant-italian" will be more similar to the state "search-restaurant-indian" than to the state "book-hotel-tonight". The statistics learnt from the "italian" dialogue state could be used to generalise to the "indian" one. In the context of VUIs developed for dysarthric speakers, two values whose associated concepts have different ASR accuracy, will have different state tracking performance. For example, if the command "TV" is recognised correctly 90% of the times and the command "radio" only 30% of the times, the state tracking statistics of the dialogue states related to these commands will be very different. If a value-specific feature able to compute similarities between dialogue states , $\mathbf{s}_v^t$, can be defined, this feature could be used to track new dialogue states using statistics learnt from the most similar states seen in the training data, improving the generalisation to unseen states.

5.3.2 Value specific features for dialogue state similarity computation

The model explained in section 2.5.3 works with value-specific general features $\mathbf{o}_v^t$ which do not help to relate dialogue states with similar state tracking behaviour. In a VUI such as hS, however, different values whose related concepts have similar ASR performance can also have similar DST behaviour. Therefore, features that give information about the ASR performance of each value can help to generalise to unseen dialogue states.

The following two subsections present two approaches to extract these features, $\mathbf{s}_v^t$. Firstly, it is proposed to use a held out set of recording to estimate the ASR accuracy related to each value (a similar approach to the one taken to compute the *APW* features in section 5.2.2). Secondly, as a held out set of recordings can be costly to obtain when the domain of the system is extended, it is proposed to extract these features directly from the phonetic structure of the commands.

ASR value accuracy features

Under the assumption that dialogue states with similar ASR or SLU accuracy will have similar state tracking behaviour, one of the best possible features to compute dialogue state similarities is an estimation of the actual ASR or SLU accuracy. Even if the accuracy can be difficult to estimate in many SDSs, in a command-based VUI such as homeService it is possible. As in hS the user interacts with the system using single word commands, the output of the ASR is an N-best list of commands, where each of the commands is associated with a value in the slot-value ontology. If recordings of the commands related to the unseen dialogue states are available

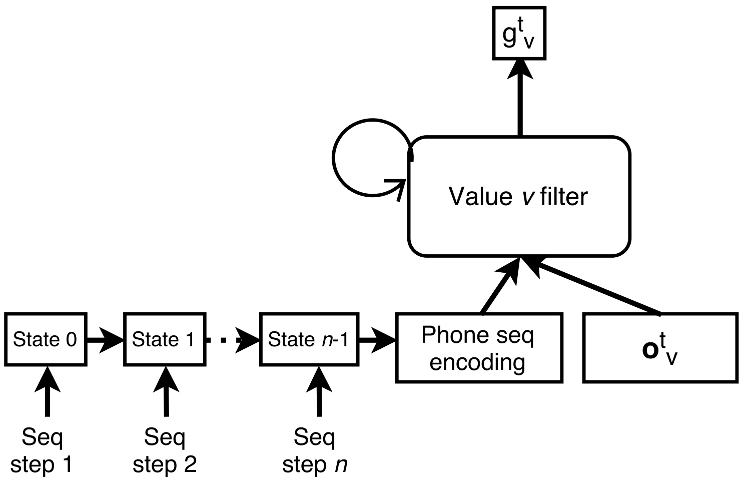

Figure 5.4: *Joint RNN encoder for a single value filter. The network in the left side represents the encoder, where the last state (phone seq encoding) represent the phone feature vector* $\mathbf{s}^t_v$.

(e.g. the enrolment data in hS), these recordings can be used to estimate the ASR performance of these commands. Then, the value specific features for each value filter, $\mathbf{o}^t_v$, can be extended by concatenating the ASR accuracy of the command associated to that value, $\mathbf{s}^t_v$. When the tracker faces a value not seen in the training data, it can improve the estimation of the probability of that value by using the statistics learnt form values with similar ASR performance.

Phone related features

The accuracy estimates proposed as features in the previous subsection need to be inferred from a held out set of word recordings. In some cases, however, this held out set may not be available. In hS, the enrolment data is collected when the system is newly set up and the system may be extended without collection of new enrolment data (see section 2.1.1). In order to avoid this requirement, the phonetic structure of the commands can be used to find similarities between dialogue states. If it is assumed that phonetically similar commands will have similar recognition rates, general features extracted from the phonetic structure of the commands can help to relate similar dialogue states. For example, the ASR can find "problematic phones" – i.e. words that contain phones or phone sequences that, due to the poor pronunciation of the user, are consistently misrecognised. Therefore, the state tracker can learn to detect such problematic phones and adapt its dialogue state inference to the presence of these phones. If an unseen dialogue state that contains these phone patterns is tracked, the state tracker can infer the probability of that state more accurately. Using the phonetic structure of the commands as additional feature for state tracking can be interpreted as moving from performing state tracking in the "command space", where similarities between dialogue states cannot be computed, to performing state tracking with the help of the "phone space", where those similarities can be estimated. In other words, the phonetic structure of the commands can be interpreted as a space composed by subunits of the commands, where similarities between states can be computed.

Phone related features can be extracted in several ways. For example, a deep neural network trained jointly with the ASR can be used to extract a sequence of phone posterior features, one vector per speech frame (Christensen et al., 2013a). Another way is to use a pronunciation dictionary (Richmond et al., 2010) to decompose the output of the ASR into sequences of phones. The latter method can also be used to extract a "phonetic fingerprint" of the associated value for each filter. For example, a filter which is tracking the value "RADIO", would have the sequence of phones [reɪdɪəʊ] as phonetic fingerprint.

In each dialogue turn, these phonetic features will be composed by sequences of different length. In the case of the ASR phone posteriors, the sequence length is equal to the number of speech frames. If a pronunciation dictionary is used, the length of the sequence will be equal to the number of phones in the command. In each dialogue turn, however, the input of the tracker needs to be a fixed length vector. Therefore, a method to transform these sequences into fixed length vectors is needed. A straightforward method is to compute the mean vector of the sequence, thereby losing the phone order information. In addition, the number of phones that the sequence has would affect the value of each phone in the mean vector. Conveniently, RNN encoders (Cho et al., 2014) have been recently shown to be able to efficiently encode sequences of arbitrary length in several machine learning tasks (Sutskever et al., 2014, Wen et al., 2016). Therefore, to compress the phone sequences in fixed length vectors while maintaining the ordering and the phone length information of the sequence, the use of an RNN encoder is proposed. In the next two subsections, two different ways to train this encoder are proposed, jointly with the model and trained in an independent pronunciation dictionary.

Joint RNN phone encoder

The state of an RNN in each step is a vector representation of all the previous sequence inputs seen by the model. Therefore, the final state after processing a sequence can be interpreted as a fixed length encoding of the sequence. If this encoding is put to the filters of the state tracker, the tracker and the encoder can be trained jointly, using back-propagation. Using this approach, the encoding of the

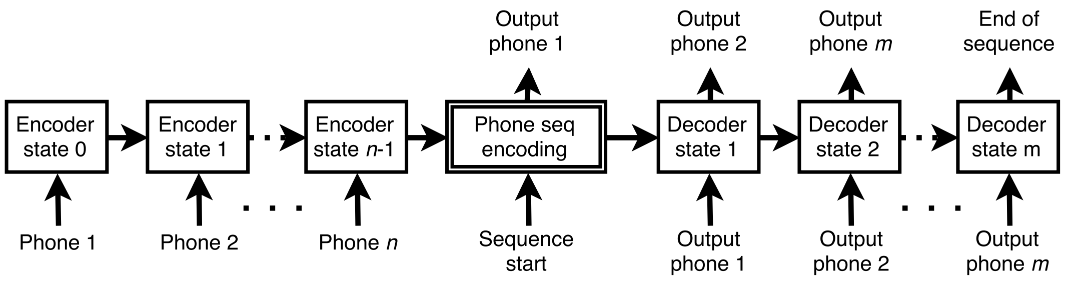

Figure 5.5: *Seq2seq phone encoder. The double-lined rectangle represents the last state of the encoder, which corresponds to the phone feature vector* $\mathbf{s}_v^t$.

phonetic sequence in each turn $\mathbf{s}_v^t$ is concatenated with the value specific features $\mathbf{o}_v^t$ for each filter as shown in figure 5.4. This defines a structure with two stacked RNNs, one encoding the phonetic sequences per turn and the other processing the sequence of dialogue turns.

Seq2seq phone encoder

Encoding the phone sequences into fixed length "dense" representations allows the computation of similarities which resemble the computing of word embeddings (Mikolov et al., 2010), where words are mapped into a continuous space in which similarities between them can be computed. The difference lies in the fact that word embeddings transform one-hot encodings[1] of words into dense vectors, while the creation of phone-based features for DST requires the transformation of *sequences* of one-hot encodings of phones into dense vectors. Sequence to sequence models (a.k.a. *seq2seq* models, RNN encoder-decoders), can be used to perform such a task. These models (shown in figure 5.5) consist of two RNNs: an *encoder*, which processes the input sequence into a fixed length vector (the final RNN state) and a *decoder*, which "unrolls" the encoded state into an output sequence. Therefore, once this model has been trained, the features $\mathbf{s}_v^t$ can be defined as the final state of the encoder (the double-lined block in figure 5.5). To train the seq2seq model, a similar approach to *auto-encoders* (Vincent et al., 2008) is taken. Following this approach, the input and target sequences used during training are the same, forcing the model to learn to encode the input sequence into a fixed length vector and then use this vector to reconstruct the original sequence.

For the experiments presented in this section, the *combilex* pronunciation dictionary (Richmond et al., 2010) has been used to train the model. The encoder

[1]One-hot encoding is a technique used to encode categorical discrete features into numerical vectors, where the size of the vectors is equal to the number of categories and each vector is composed by zeroes except for a one in the position of its corresponding category.

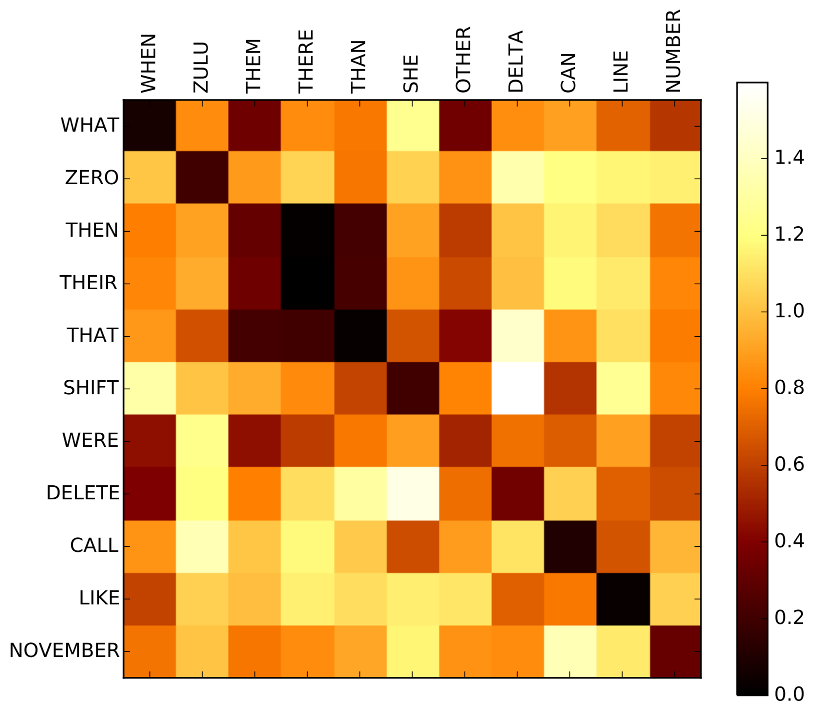

Figure 5.6: *Cosine distance in the phone encoding space between different words of the UASpeech database.*

and decoder RNNs of the *seq2seq* model are composed by two stacked layers of 20 LSTM units each. When this model is trained in two thirds of the phone sequences of the combilex dictionary, it is able to reconstruct more than 95% of the remaining third of phone sequences correctly. When this same model (trained in combilex data) is used in the dysarthric pronunciation dictionary composed by words of the UASpeech database (Christensen et al., 2012a), the model is able to reconstruct more than 90% of these phone sequences. This means that the model is able to compress sequences of one-hot vectors of size 45 (the number of phones in US English) into vectors of size 20 which contain most of the key information of the original phone sequence. To show if the mappings done by this model effectively relate similar phone sequences, figure 5.6 shows the cosine distance between the dense phone representations of two sets of words of the UASpeech database. The results observed in this figure illustrate that in most of the cases these encodings are able to effectively relate words with similar phone composition. For example, the dense representation of the word "their" is close to words such as "them", "there" and "that", and far from "she", "can" and "line".

5.3.3 Experimental setup

To evaluate the methods proposed in section 5.3.2 to improve generalization to unseen states, these methods are tested in a RNN-LSTM based general state tracker using different value specific features. To simulate the scenario in which a user changes the devices to control at home, data is collected in two simulated homeService environments which use a different set of commands to control the devices, named *domain A* and *domain B* (see section 5.1). Then, the trackers are trained with dialogues collected in domain A and tested in dialogues from domain B. To evaluate the ability of these features to generalise to infrequently seen states, a small amount of dialogues from domain B is included in the training data in a second set of experiments.

General RNN-LSTM based state tracker parameters

For each of the 12 speaker-ASR pairs (corresponding to the low and mid intelligibility speakers in the UASpeech database interacting with a non-adapted and adapted ASR), a general dialogue state tracker based on the model described in section 2.5.3 has been trained. Each value filter is composed by a linear feedforward layer of size 20 and an LSTM layer of size 30. In order to reduce overfitting, *dropout* regularisation is used, with a dropout rate of 0.2 in the input connections and 0.5 in the remaining non-recurrent connections. The models are trained for 60 iterations with stochastic gradient descent. For each of the 12 speaker-ASR pairs, each state tracker is trained on 1000 dialogues from that speaker, collected in domain A (with a 0.8-0.2 train-validation split) and is evaluated on 1000 dialogues from that speaker, collected in domain B. The validation set is used to choose the parameter set corresponding to the best iteration. Model combination is also used to avoid overfitting. Every model is trained with three different seeds and five different parameter sets are saved for each seed, one for the best iteration in the first 20 and then another for the best iteration in each interval of 10 iterations. This approach is taken because, depending on the features $\mathbf{s}_v^t$ used, the models tend to overfit after a different number of iterations. By combining the best parameter sets of each interval, the risk of selecting similar parameter sets, all corresponding to overfitted models, is reduced.

ASR and phone related general features

In each turn t, each value-specific state tracker (filter) takes as input the value-specific features. In the hS system evaluated in this section, these features $\mathbf{o}_v^t$ are composed by the concatenation of the following elements:

- The confidence score of the command related to the value v seen in the ASR N-best output.
- The confidence scores of the meta-commands "yes" and "no".
- A (value-specific[1]) one-hot encoding of the last system action.

In addition, the models are evaluated concatenating the value specific features $\mathbf{o}_v^t$ with the following ASR and phone related general features $\mathbf{s}_v^t$:

- *ValAcc*: As mentioned in section 5.3.2, the ASR performance estimation of the command corresponding to the value of the filter can be used as general feature. In the following experiments, the accuracy per command is used, defining $\mathbf{s}_v^t$ as the ASR accuracy of the value v estimated in a held out set.
- *PhSeq*: As explained in section 5.3.2, a weighted sequence of phones can be generated from the ASR output (N-best list of commands), using a pronunciation dictionary. The pronunciation dictionary for words of the UASpeech database described in Christensen et al. (2012a) is used to translate each command into a sequence of one-hot encodings of phones (the size of the one-hot encoding is 45, like the number of phones in US English). Each of these encodings is weighted by the confidence score of its corresponding command in the N-best list. The sum of these weighted sequences is then fed into an RNN encoder (figure 5.4) and $\mathbf{s}_v^t$ is defined as the vector corresponding to the final state of this RNN. The RNN is composed by a single *GRU* (Chung et al., 2014) layer of size 15.
- *PostSeq*: As previously mentioned in section 5.3.2, a deep neural network can be used to extract the posterior probabilities of the phones during the ASR step. For the following experiments, a sequence of vectors (one vector per speech frame) with monophone-state level posterior probabilities are extracted from the output layers of a deep neural network trained on the UASpeech corpus (this approach is taken from Christensen et al. (2013a)). The extracted vectors contain the posteriors of each of the three states (initial, central, and final) for the 45 phones of US English. To reduce the dimensionality of the vectors, the posteriors of the three states of each phone are merged by summing them. To reduce the length of the sequence, the mean of each group of 10 speech frames is taken. This produces a sequence of vectors of size 45 and maximum length of 20, which is fed into an RNN encoder in the same way as *PhSeq* features to obtain $\mathbf{s}_v^t$.

[1]For actions related to specific values, e.g. "confirm:TV", this encoding indicates if the value being confirmed is the value of the filter.

	Joint	Slot 1	Slot 2	Slot 3	Mean
Baseline	50.5%$^-$	81.0%$^-$	51.5%$^-$	55.7%$^-$	62.7%$^-$
General	68.8%	87.5%	66.5%	67.6%	73.9%
ValAcc	74.1%$^+$	88.9%$^+$	72.1%$^+$	66.5%	75.8%$^+$
PhSeq	68.3%	89.3%$^+$	66.2%	67.7%	74.4%
PostSeq	67.9%	89.2%$^+$	65.9%	67.6%	74.2%
ValPhEnc	57.9%$^-$	77.9%$^-$	61.5%$^-$	59.3%$^-$	66.2%$^-$
PhSeq-ValPhEnc	58.5%$^-$	79.8%$^-$	62.0%$^-$	58.9%$^-$	66.9%$^-$

Table 5.2: *Joint, mean and per slot state tracking accuracy (columns) of the trackers trained on domain A and tested on domain B for trackers using different features (rows). +/- indicates statistically significantly better/worse than the General tracker ($p < 0.01$), computed with a two-proportion z-test.*

- *ValPhEnc*: As also explained in section 5.3.2, a "value fingerprint" can be extracted for each value using a pronunciation dictionary. For each value filter, $\mathbf{s}_v^t$ is defined as the 20 dimensional encoding of the sequence of phones of the command associated with the value v extracted from the *seq2seq* model defined in section 5.3.2. The encoder and decoder RNNs of the seq2seq model are composed of two layers of 20 LSTM units. The model is trained on the *combilex* dictionary (Richmond et al., 2010).

Note that two different kinds of features can be distinguished: *value identity* features and *ASR output* features. Value identity features (*ValAcc* and *ValPhEnc*) give information about the value tracked by each filter. These features are different for each filter (as each filter has a different associated value), but are time invariant (they do not change over turns). These features can be used to relate dialogue states similar in the phone space. ASR output features (*PhSeq* and *PostSeq*), on the other hand, give information about the ASR output observed. They are the same for each filter but change in each dialogue turn. These features can be used to detect phone patterns occurring in the ASR output (e.g. problematic phones).

5.3.4 Experimental results

In this section, the results obtained from the state tracking experiments, when using the different general augmented features proposed in section 5.3.3, are presented. To be sure that the proposed features work independently of the set of commands selected, different models are trained and tested using the three randomly selected vocabularies of 72 words (see section 5.1). For each of the three vocabularies and for each of the 12 speaker-ASR pairs (corresponding to the six test speakers interacting with a non adapted and a fully adapted ASR), a general state tracker is trained with data collected from that speaker-ASR pair using that vocabulary. Each presented result shows the mean for the three vocabularies and for the 12 speaker-ASR pairs. For each evaluated augmented feature, the joint state tracking accuracy, the accuracy of each individual slot and the mean accuracy of the three slots is presented. These five values are presented because it was found that the relation between the mean slot accuracy and the joint accuracy is highly non-linear, due to the high dependency on the ontology of the joint goals. When joining the slot outputs (equation 5.1), the "invalid goals" are discarded, but the cost optimised is related to the mean accuracy of the slots. Therefore, the joint accuracy metric does not reflect the cost that the models are minimising. Future work should explore methods to join the slot outputs more efficiently or to train the models to maximise the joint accuracy.

Table 5.2 presents the state tracking accuracy results for the model described in section 5.3.3, using only value specific general features (*General*) and using the different augmented features proposed in section 5.3.3. To evaluate the generalisation to unseen dialogue states, the models are trained on data from *domain A* and evaluated in data from *domain B*. *Baseline* presents the state tracking accuracy for the same rule-based state tracker used in section 5.2.3 as baseline. First, observe that the machine learnt state trackers perform considerably better than rule-based ones, with *General* outperforming *Baseline* by more than 10% in mean slot accuracy. Including the ASR accuracy estimates (*ValAcc*), outperforms all the other approaches, performing 5.2% better than *General* in the joint goal accuracy and 1.9% better in the mean slot accuracy. Including *PhSeq* or *PhSeq* features slightly degrades the performance in the joint goal but outperforms the *General* features in the mean slot accuracy by 0.5% and 0.3% respectively. *ValPhEnc* and *PhSeq-ValPhEnc* features, however, perform much worse than the other features. A detailed examination of the training results showed that, compared to *General* features, these features were performing about 10% better in the validation set (*domain A*), while getting 10% worse results in the test set (*domain B*). This suggests that the model is overfitting to the training data.

In order to evaluate if these features can help to generalise to infrequently seen dialogue states, table 5.3 shows the accuracy results when a small number of dialogues from *domain B* (200 dialogues) are included in the training data. The mean accuracy gain obtained by including these dialogues in the training data is very small for *General* and *PhSeq* features, performing 0.6% and 0.2% better respectively. *ValPhEnc* features, however, show a large improvement, outperforming *General* features by 4% in the joint goal and by more than 5.4% in the mean slot accuracy. This improvement is seen in all the slots individually. To ensure that

	Joint	Slot 1	Slot 2	Slot 3	Mean
General	68.9%	87.8%	66.9%	67.5%	74.0%
ValAcc	74.6%$^+$	89.6%$^+$	72.7%$^+$	67.2%	76.5%$^+$
PhSeq	69.2%	89.5%$^+$	66.1%	68.2%	74.6%
ValPhEnc	72.9%$^+$	89.8%$^+$	72.8%$^+$	75.6%$^+$	79.4%$^+$
PhSeq-ValPhEnc	73.4%$^+$	91.6%$^+$	74.0%$^+$	77.2%$^+$	80.9%$^+$
ValId	60.8%$^-$	86.3%$^+$	66.9%	75.0%$^+$	76.1%$^+$

Table 5.3: *Joint, mean and per-slot state tracking accuracy (columns) of the trackers when including 200 dialogues from domain B in the training data for trackers using different features (rows). +/- indicates statistically significantly better/worse than the General tracker ($p < 0.01$), computed with a two-proportion z-test.*

the model is not just using the *ValPhEnc* features to learn the identities of the words, *ValId* features extend the value specific features $\mathbf{o}_v^t$ by concatenating a one-hot encoding of the value identity to them. Including these features improves the mean slot accuracy with respect to *General* by 2% (even if the performance in the joint goal decreases). The performance using these features, however, is still more than 3% below the *ValPhEnc* features. If the concatenation of *PhSeq* and *ValPhEnc* features is used, the mean slot accuracy outperforms all the other features, performing 6.9% better than *General* and 4.4% better than *ValAcc* features.

Experiment conclusions

Several conclusions can be drawn from the results presented above. First, it can be seen that the baseline tracker is outperformed by more than 10% by the *General* tracker. This baseline tracker is the same one used in the experiments presented in section 5.2.3, where it had performed much better. However, the ASR accuracy in the experiments done in this section is lower, because the size of the ASR vocabulary is 72 instead of 36. This suggests that the baseline tracker does not perform well in environments with a more challenging ASR. In addition, the vocabulary changed, so the baseline tracker might be dependent on the vocabulary used. On the other hand, note that the performance of the machine learnt trackers presented in this section is better than the ones presented in section 5.2.3. This might seem counter intuitive, but the trackers in that section were trained with data from other speakers while the trackers in this section are trained with data from the same speaker, and the vocabularies used in each section are different, so the results are not directly comparable.

As it was expected, including the accuracy estimates (*ValAcc*) outperformed all the other approaches, especially for the joint goal. This is because, if the assumption that values with similar ASR accuracy have similar state tracking performance is true, these features are the best estimate to compute similarities between states. The improvement obtained from *PhSeq* features, however, was smaller. Comparing the slot by slot results, it can be seen that the accuracy increase comes from slot 1, where *PhSeq* features outperform *General* features by almost 2%. In the other two slots, the accuracy is similar. This suggests that the performance of these features could be related to the values of the slots. Analysing the performance of *PostSeq* features, it can be seen that is very similar to the performance of *PhSeq*, suggesting that these two features carry very similar information.

In contrast to the other augmented features, using *ValPhEnc* degraded the performance of the tracker, due to strong overfitting to the training data. This might be caused by the size of the vocabulary (36 words in each domain), which is not large enough for the model to find similarities between the phone encoding vectors. In other words, there is not enough overlap between the phones in the different domains; therefore, these feature vectors mostly carry meaningless information. However, when a small number of dialogues from domain B was included in the training data, using *ValPhEnc* features and especially the concatenation of *ValPhEnc* and *PhSeq* features, increased the tracker's performance by a large amount. These features performed even 4.4% better than *ValAcc* features, which suggests that they carry more useful information than just the ASR accuracy estimate. These features were compared with *ValId* features to show that the model is not just learning the identity of each value. Therefore, the results suggest that *ValPhEnc* features are effectively correlating the state tracking performance of values similar in the phone encoding space to help to generalise to infrequently seen states. Using a larger corpus with more phone variability to train the models could make these features useful to generalise to unseen states too.

5.4 Conclusions

This chapter has investigated how augmenting the dialogue features typically used by a dialogue state tracker, can improve the generalisation in scenarios of mismatched training and evaluation data. In an RNN-LSTM based state tracker personalised to a target speaker, speaker specific features extracted from the raw acoustics and from the ASR showed an improvement in generalisation when the model was trained with data from other source speakers. It was also shown that the improvement obtained with speaker features is larger when small amounts of data from the target speaker become available, suggesting that the approach is useful to generalise to unseen and infrequently seen speakers. In the case of a general state tracker evaluated on a different domain, extending the value specific features with ASR accuracy estimates showed to improve the state tracking accuracy. The performance when using dense representations of phone sequences encoded with a *seq2seq* model decreased due to strong overfitting to the training data. The most probable reason is the small variability of the command vocabulary (36 commands in each domain), which is not large enough for the model to find useful correlations between phone encodings. However, when a small amount of data from the unseen domain was included into the training data, phone encodings greatly increased DST accuracy. This suggests that phone encodings are useful as dense representations of the phonetic structure of the command, helping the model correlate state tracking performance of values close in the phonetic encoding space.

The experiments performed in this chapter showed that, feature-rich discriminative DST, opens up the possibility of using numerous different features which can give more information than the information present in the SLU output. In this chapter, only a small amount of the possible useful extended features have been studied and only in two different scenarios. The speaker generalisation features have been tested in a VUI designed for dysarthric speakers, but these features have the potential to be used with normal speakers too. For example, speaker features can help to relate speakers with similar accents. In addition, these features seem to work better when used with larger corpora. When working with "normal" speakers, having access to data from more source speakers would be possible, which could in turn increase the chance of finding speakers "similar" to the target. This would increase the effectiveness of the features. In addition, DST input feature augmentation could be potentially applied to many other domains where mismatch between the training and testing conditions exist. For example, a model could be used to extract a feature indicating the amount and type of environment noise present during a dialogue. Then, the state tracker could use these features to learn to infer the dialogue state in different noise conditions.

In the case of features that help to generalise to unseen states, the proposed approach has been tested on a single-word command-based environmental control interface, where slot-value accuracies can easily be estimated. In addition, in this domain, the sequences of phonetic features are usually short. However, the same approach can be adapted to larger spoken dialogue systems by estimating the concept error rate of the SLU output of the concepts related to each slot-value pair. Longer phonetic feature sequences can also be used to detect "problematic phones", or to correlate sentences with similar phonetic composition. The main problem faced by these features was overfitting, mostly due to the small variability of the training dataset. As the size of datasets used to train machine learning models is increasing in almost every field, the release of larger dialogue corpora could increase the effectiveness of these features.

It is also worth mentioning that figure 5.6 showed that, in a similar way to word embeddings, a *seq2seq* model can be used to map sequences of phones into a vector space. In this space, cosine distances effectively computed similarities between phone sequences of possibly different lengths. Using this approach to create "phone sequence embeddings" could have potential applications in other domains, such as language modelling or named entity recognition.

In general, it is reasonable to think that the typical pipeline-based dialogue system architecture will gradually include more and more co-dependencies between modules. As well as the dialogue state tracker can benefit from having access to features coming from the ASR, the ASR and SLU modules can benefit from having access to the dialogue context in the form of the belief state (Jonson, 2006). For example, it is more likely to recognise the word "thai" than the word "they" if the dialogue system just asked about the type of food you are interested in. As the size of the corpora and the computing capability increase, it is possible that these codependencies between the modules will eventually lead to end-to-end spoken dialogue systems: systems composed by a single module that takes an acoustic signal as input and produce a synthesised spoken answer as output.

Chapter 6

Conclusions

The emergence of voice user interfaces to control personal devices such as smartphones, suggests that the personalisation of these interfaces will be an important research topic in the following years. Before the work done in this thesis, however, very little attention has been paid to dialogue management adaptation to specific speakers or environments, especially for POMDP-based dialogue managers. This thesis has identified and addressed the problems that have to be taken into account when a POMDP-based dialogue manager is used by a single user over a long period of time, in the context of VUIs developed for users with speech disorders. Several modifications to state-of-the-art POMDP models and algorithms have been developed, showing that they can improve the performance of VUIs personalised to dysarthric speakers working in varying environments. The techniques developed, however, are not specific to dysarthic speakers and can be potentially applied to any kind of personal dialogue manager.

DM is a very wide research topic in where many different modules work together to control the dialogue flow (e.g. state tracker, policy, reward modelling, turn taking...). As investigating the personalisation of every element of a dialogue manager is infeasible, this thesis has only explored the personalisation of the two key DM modules – the dialogue policy and the dialogue state tracker. As none or very little research has been done on this topic before, the techniques can probably be improved, but the main contribution of this thesis is to show that the personalisation of DM models can improve the dialogue performance, establishing a research topic with a lot of potential.

6.1 Thesis summary and achievements

Adaptation of POMDP-based dialogue management models to specific speakers or environments can be considered a very immature topic. This thesis has identified several problems that will be encountered when developing a personalised POMDP-based dialogue manager and proposed solutions for these problems.

Chapter 3 explored the performance of the POMDP DM framework applied to a dialogue domain different from the habitually studied ones – VUIs for users with speech disorders. Spoken interfaces can have a great impact in the life quality of users with restricted upper limb mobility, but if these users also suffer from dysarthria, the poor ASR performance of the speaker independent interfaces reduces the utility of the system. Personalisation of the DM models of these interfaces can greatly improve the interaction performance. Even if severe dysarthric speakers cannot be understood by unfamiliar speakers, their family and carers learn to understand them by interacting with them, adapting to their communication characteristics. The carers learn to understand the dysarthric speakers gradually: they go through an adaptation and learning process, where the carer learns to understand the speaker by trial and error, until the carer is able to communicate efficiently with the speaker. This learning strategy resembles the reinforcement learning framework, suggesting that POMDP-based DM is a very suitable framework for these kind of interfaces. Systems developed for severely dysarthric speakers need to be easy to personalise and highly adaptive, and the POMDP framework provides those characteristics.

Chapter 3 continued by developing a VUI environment based on the homeService system (Christensen et al., 2013b), as well as a set of simulated dysarthric users to evaluate it. A tree-based dialogue state representation for these interfaces was proposed, making the application of model-based reinforcement learning algorithms tractable. This tree-based representation used the POMDP framework to increase the robustness against high ASR error rate, while sacrificing the mixed initiative interaction ability to maintain tractability. In section 3.4.2, it was shown that the POMDP framework can increase the interaction performance in VUIs for dysarthric speakers, especially when the ASR is not adapted or adapted with a small amount of data. In addition, the experiments done in sections 3.1.2 and 3.4.2 identified two main differences between long-term personal interaction with a SDS and short term interaction based SDSs: the variability between different speakers and the environment change over time (in the form of an ASR adapted with increasing amounts of data). It was concluded that to deal with these two aspects, existing POMDP models needed to be adapted.

In chapter 4, the two aspects identified in chapter 3 were analysed in greater depth, identifying both of them as RL problems where the training data comes from different (but possibly related) distributions. Then, in section 4.1, two RL models able to work in these situations were developed. These models are based on state-of-the-art, model-free, GP-RL models ((Gašić and Young, 2014)), which are tractable for real-world sized SDSs but do not have a practical mechanism to deal with data coming from different distributions. The most promising of the two GP-RL model variations developed is based on a *temporal difference kernel*. This approach applies the temporal difference operation inside the kernel of a GP, defining the co-dependencies between consecutive dialogue turns more strongly. In addition, the TD kernel framework "normalises" the GP-RL equations to the same shape as the equations of GP regression models, simplifying the understanding of the GP-RL framework and opening the possibility of easily applying techniques used in GP regression (e.g. sparsification (Quiñonero-Candela and Rasmussen, 2005), hyperparameter optimization (Rasmussen, 2006), GPLVM (Lawrence, 2004)). This framework was shown to be applicable to transfer learning scenarios likely to be found in hS systems: a variable environment scenario; and the initialisation of a personalised dialogue policy with data from other speakers.

In addition, in sections 4.2.4 and 4.3.1 a set of "environment" features were proposed for each of these two scenarios, giving information about the different environments (distributions) in which each data point was collected. In figures 4.2 and 4.6, the combination of the TD kernel based model with the environment features was shown to improve the dialogue interaction performance in these two scenarios. In the case of initialising a personalised dialogue policy with data from other speakers, the TD model was shown to be suitable for online policy optimisation. In figure 4.3 it was demonstrated that, using this model, initialising the policy with data from other speakers greatly increases the initial performance of the policy and needs far fewer interactions to achieve a near-optimal performance.

In chapter 5 personalisation techniques were investigated for dialogue state tracking, a research area never explored before. Due to the ability of discriminative dialogue state trackers to use high dimensional, possibly correlated, input features, the environment features proposed in chapter 4 were found to have more potential to be used in the DST step. A dialogue architecture alternative to the typical "pipeline" architecture was proposed, including information extracted from the acoustics and from the ASR into the DST step. In table 5.1, it was shown that using speaker related environmental features can help discriminative state trackers to better generalise to speakers not seen in the training data, obtaining an absolute improvement of more than 3.6% in dialogue state tracking accuracy. Domain extension for personalised dialogue state trackers was also studied in section 5.3, proposing to use ASR and phone related features to find similarities between dialogue states. The results presented in tables 5.2 and 5.3 showed that the use of these features improves the generalisation to unseen or infrequently seen states. In addition, this chapter showed the potential of using recurrent neural network encoders as feature extractors. The research done in this chapter showed that including richer and more "raw" features in the DM step of a SDS improves the performance, and could gradually lead to less dependence on the typical pipeline architecture, eventually leading to end-to-end SDSs.

6.2 Limitations and future work

Even if the methods developed in this thesis define a new state of the art in personalised POMDP-based dialogue management, the work has some limitations which should be mentioned. Future work related to this thesis should take these limitations into account and try to overcome them.

Firstly, SDSs or VUIs are very complex systems composed of several modules that work together in order to control the dialogue flow. All these modules are co-dependant; the performance of each module depends on the performance of the rest. However, each module is usually researched independently, and specific evaluation metrics have been developed to test the performance of each of them. Following this approach, in this thesis, most of the techniques proposed are evaluated in "isolated" environments. For example, in chapter 4, the environment features are either tested in ASR-changing environments or in speaker dependent environments, but the techniques are not tested in the combination of these two environments. In chapter 5, the performance increase obtained in the DST is not tested in a whole dialogue system to see its impact in the overall dialogue performance. As the techniques developed in chapters 4 and 5 can be complementary, in future work it would be interesting to evaluate these techniques working together in a full scale VUI.

In a similar way, chapter 3 presented a tractable tree architecture representation for the dialogue state. This is a very interesting approach, taking advantage of the knowledge about the structure of the VUI to reduce the number of parameters that need to be learnt. In order to use this dialogue representation, however, several assumptions had to be made. Following the trend that research in machine learning is taking, where larger and larger datasets are being used to train the models, the limitations imposed by this structure and its related assumptions could decrease the utility of the tree-based dialogue representation when larger datasets are used for training. In the future, it would be interesting to evaluate the models proposed in chapter 4 in a slot-based dialogue state representation architecture.

In addition, another limitation of this thesis is that it has not tested the developed models with real users. However, due to the novelty of the research done (in two aspects: personalised dialogue management and POMDP framework for dysarthric users), most of the approaches had to be developed almost from scratch, with no other models to compare with. In the homeService project, where apart from the ASR, the rest of the modules follow simple hand-crafted rules, the cost of testing the systems with dysarthric users turned out to be extremely high (Nicolao et al., 2016). Developing more complex statistical interaction models requires a lot of evaluation, model variations, hyperparameter tuning, etc. Therefore, the cost of testing these models with real dysarthric users would have been impractical. Furthermore, the main objective of this thesis was to establish whether the POMDP-DM framework is a suitable approach for personalised VUIs for dysarthric speakers. Even if simulated users cannot perfectly evaluate the performance of this framework, they can give an estimate of the performance and compare the most promising models. In low-resourced domains such as dialogue with dysarthric users, the development of more sophisticated and accurate simulated users could greatly help academic research and future work should focus on that.

Finally, chapter 4 and 5 showed the potential of using extended input features to control the dialogue flow. GP-RL based models, however, could not make use of all the potential information included in these features. Moreover, discriminative trackers could model the correlations between different dimensions of the extended input features better, but the models are optimised with respect to the dialogue state classification accuracy, instead of using the actual dialogue performance (e.g. the dialogue reward). Therefore, the dialogue state representation needs to be designed manually and a training corpus of dialogues with annotated states is required. Recently, policy gradient methods (Fatemi et al., 2016, Su et al., 2016a, Sutton et al., 1999) have been shown to be a promising neural network based approach to policy optimization. Interestingly, the gradient used to optimise these models can be back-propagated through the DST network, allowing the state tracker and the policy being learnt jointly (Mnih et al., 2015, Silver et al., 2016). Using this joint framework in combination with the proposed environment features (or any other environment feature) would give the model more freedom to learn the correlations between the input features, leading to a better overall dialogue performance.

6.3 Future directions in personalised dialogue management

Spoken dialogue management is a rapidly developing research area. Ten years ago, dialogue management models that scaled to real world sized SDSs were developed (Jurčíček et al., 2010, Thomson and Young, 2010, Williams and Young, 2007). These models, however, needed thousands of dialogues to learn a useful policy. Hence, a lot of the research was focused on building good simulated users to be able to train reinforcement learning models (Georgila et al., 2005, Schatzmann et al., 2006, 2007a). Five years ago, the main research focus shifted to finding reinforcement learning algorithms that could learn dialogue policies in the least possible interactions, to be able to be trained by directly interacting with real users (Gašić et al., 2011, Geist and Pietquin, 2010). In 2013 the DST challenges began (Henderson et al., 2014a,b, Williams et al., 2013), which gave a test-bed for comparison of different models in a easier way than interacting with real users. Thus, most of the research focus on dialogue management shifted to dialogue state tracking (Henderson et al., 2014c, Lee, 2013, Williams, 2014). Nowadays, building systems that can be trained and work in multiple different domains (a.k.a. multi-domain systems) are attracting a lot of attention (Gašić et al., 2015, Hakkani-Tür et al., 2016, Mrkšić et al., 2015). The interest in neural network based policy models is also increasing (Fatemi et al., 2016, Su et al., 2016a). Ironically, these models need large amounts of data to be trained and are thus contradicting the research trends from five years ago, when models which needed the least amount of data to be trained were researched. Using network based policies, it is also possible to back-propagate the errors from the rewards to the state tracker, thus opening the possibility of developing end-to-end dialogue managers which would not need to design and train a state tracking model (because the network itself can learn the shape and weights of the state tracker automatically from raw inputs (Mnih et al., 2015, Silver et al., 2016)).

What most of the speech technology community seems to agree on, is that the next challenge for artificial intelligence and machine learning is to build systems that are able to understand human language and communicate with them in a human-like way. It seems reasonable to think that research in spoken dialogue management will play a crucial role in this challenge[1]. It is still not clear whether the POMDP dialogue management framework is the most suitable one for this task, since even though it was proposed almost 20 years ago, commercial dialogue systems still rely on rule-based frameworks to manage the dialogue interaction.

[1]It is also worth mentioning that, even if this thesis has focused only on spoken dialogue management, the interest in text based dialogue management is also rapidly increasing in the form of chat-bots or conversational agents.

Nevertheless, the reinforcement learning framework highly resembles the way humans learn to interact, suggesting that it is a good theoretical basis for statistical dialogue management modelling. It is possible that the lack of success in implementing real world systems obtained by the POMDP framework so far is related to the difficulty and expense of collecting dialogue data, and the difficulties faced by current models to use data collected in different dialogue domains.

Nevertheless, some of the most important tech companies (e.g. Google, Apple, Amazon...) are investing a lot of resources[1] in dialogue management and language understanding research, and several startups have been created related to these topics. The main commercial interest of these companies is the development of general purpose, high performance, personal assistants. These assistants aim to change the way users interact with machines, possibly removing the typical physical interfaces (e.g. touchscreen or mouse) or making the users less reliant on them. Personal assistants can change the way users shop, study, communicate with friends, plan their agendas, etc. These assistants can also improve the search for information on the web, letting users navigate the web by asking sequences of questions to the personal assistants. As these assistants will be likely used by a single user for a prolonged period of time, it is reasonable to think that dialogue management personalisation will be a very important part of research on personal assistants.

After the remarkable success obtained by deep neural networks in supervised learning tasks (Bengio, 2009, Hinton et al., 2012, Krizhevsky et al., 2012, Mesnil et al., 2013), building reinforcement learning methods based on deep neural networks seems to be the next challenge faced by machine learning[2]. The reinforcement learning framework is more similar to the way that humans learn, so it could be considered closer to "true" artificial intelligence. While, supervised learning resembles a student learning about a specific topic from a teacher, reinforcement learning resembles the way humans (and other species) learn by direct interaction with their surroundings. Deep reinforcement learning has already achieved some remarkable successes such as playing Atari games on an human expertise level (Mnih et al., 2015) or beating the word champion of the game Go (Silver et al., 2016), a task previously believed to be too complex to be performed by a computer. These tasks, however, are still defined in very narrow domains, where the computer can be exposed to thousands or even millions of interactions with the

[1]The *Amazon Alexa Prize* competition (`https://developer.amazon.com/alexaprize`) is an example of the resources invested by tech companies on dialogue management, giving a total of two and a half million dollars in prizes to the teams building the dialogue systems able to interact in the most human-like way.

[2]For example, Facebook AI research has developed a "wild" environment, where algorithms that learn from raw information signals (streams of bits) can be evaluated (Sukhbaatar et al., 2015).

environment at almost no cost. When the system interacts with a more realistic environment (e.g. the real world), the number of possible interactions will be constrained by the laws of physics, so the amount of data the system has access to will be reduced. However, if a large number of agents are able to interact with the real environment at the same time, these agents could share the knowledge they obtain to speed up the learning process. This could be seen as a kind of "collective intelligence".

All these research directions and the achievements obtained in deep reinforcement learning are leading the DM field towards building end-to-end dialogue managers. End-to-end dialogue managers are not factorised into a dialogue state tracker and a policy model. They take the output of the spoken language understanding system (and any other auxiliary features) as input, and output a machine action. Because policy gradient methods can back-propagate the error up to the input of the dialogue manager, a model that does state tracking and policy optimisation could be trained jointly. The advantages of this approach are several: the need to define the labels of the state tracker would not exist, thus requiring less annotated data; the state tracker optimization would be done properly, using the reward signals instead of maximising the state classification rate; and there would not even be the need to define an ontology based dialogue state representation, because a neural network would be able to learn the structure by itself. Eventually, the SLU could also be integrated into this structure, learning jointly the policy, state tracker and SLU modules (Henderson et al., 2014d). In the end, end-to-end spoken dialogue systems, where the system directly takes the raw acoustic signal as input and outputs the machine action (in the form of a synthesised answer or in any other way), seems to be the probable direction that spoken dialogue systems will take.

End-to-end models, however, will need very large amounts of data to be trained, which possibly posses a challenge for academic research. The lack of data is already a problem in academic research on dialogue management because cheap ways to train and evaluate the models by interacting with real users do not exist. The Amazon mechanical turk has reduced the cost of interactions with real users (Jurčíček et al., 2011), but it is still expensive compared to training and evaluating supervised learning tasks, which can be done with a data corpus. On the other hand, companies have access to very large amounts of dialogue data, either because they invest a substantial amount of money to obtain it or because they collect the data through interaction of the users with their products. In the case of dialogue management for specific groups (such as dysarthric speakers), the problem is greater, as there is less available data and obtaining it is more expensive.

Even if the interest in simulated users decreased some years ago[1], direct research with real users and real data seems to be in the reach of companies with enormous resources only. The main flaw of the currently existing simulated users is that, if they are trained from data, they are just able to learn how to interact in the dialogue situations seen in the training data. To account for more variability, mechanisms to increase the dialogue situations where the simulated users can interact must be implemented, but this can be difficult to do while maintaining the coherence of the dialogue. In other areas such as speech recognition and artificial vision, however, using data augmentation techniques to increase the amount of training data has shown to improve the results, especially in low resourced domains (Ragni et al., 2014). More intense research on more realistic simulated users would greatly reduce the costs of research in dialogue management.

[1]It was argued that, if a perfectly real model of the user was known, the dialogue management model could be inferred from it without the need of interactions between the simulated user and the dialogue manager.

Appendix A

Example homeService dialogues

For a more detailed understanding of the types of dialogues that can occur in the homeService environment, this appendix presents several dialogue examples. The example dialogues are divided in two categories, dialogues interacting with the system using the hierarchical tree-based architecture presented in section 3.2.2 and the slot-based architecture presented in section 3.2.2. The examples present different dialogue situations, such as failed dialogues or recovery strategies taken by the DM. In each example dialogue, the second column represents the actions taken either by the user (user commands) or by the system (machine actions).

A.1 Dialogues in the tree-based dialogue state representation architecture

This section presents three different example dialogues using the tree-based dialogue state representation architecture. The first dialogue is an example of successful dialogue, the second is and example of failed dialogue and the third an example in where the dialogue manager is able to recover from a misunderstanding error.

Dialogue 1: Goal = *TV-Channel-One*
Dialogue starts in node "Devices"

Sub-dialogue "Devices"

User	*TV* (Speaks the command "TV")
System	*Ask* (Requests the user to repeat his last command)
User	*TV* (Repeats his last command)
System	*Node-TV* (Dialogue transitions to node "TV")

Sub-dialogue "TV"

User	*Channel* (Speaks the command "Channel")
System	*Node-Channel* (Dialogue transitions to node "TV-Channel")

Sub-dialogue "Channel"

User	*One* (Speaks the command "One")
System	*Node-One* (Performs action *TV-Channel-One*)

As an action has been taken in a terminal node, the dialogue ends. As the action taken in the terminal node matches the goal, it is a successful dialogue.

Dialogue 2: Goal = *Bluray-On*
Dialogue starts in node "Devices"

Sub-dialogue "Devices"

User	*Bluray* (Speaks the command "Bluray")
System	*Node-Bluray* (Dialogue transitions to node Bluray)

Sub-dialogue "Bluray"

User	*On* (Speaks the command "On")
System	*Ask* (Requests the user to repeat his last command)
User	*On* (Repeats his last command)
System	*Node-Off* (Performs action *Bluray-Off)*

As the action taken in the terminal node does not match the goal, the dialogue ends as a failed dialogue.

Dialogue 3: Goal = *TV-Volume-Up*
Dialogue starts in node "Devices"

Sub-dialogue "Devices"

User	*TV* (Speaks the command "TV")
System	*Node-Light* (Dialogue transitions to node Light)

Sub-dialogue "Light"

User	*Back* (Requests to go to the previous node)
System	*Node-Back* (Dialogue transitions to node Devices)

Sub-dialogue "Devices"

User	*TV* (Command "TV")
System	*Ask* (Requests the user to repeat his last command)
User	*TV* (Repeats his last command)
System	*Node-TV* (Dialogue transitions to node TV)

Sub-dialogue "TV"

User	*Volume* (Speaks the command "Volume")
System	*Node-Volume* (Dialogue transitions to node TV-Volume)

Sub-dialogue "TV-Volume"

User	*Up* (Speaks the command "Up")
System	*Node-Up* (Performs action *TV-Volume-Up*)

As the action taken in the terminal node matches the goal, the dialogue ends as a successful dialogue. The system was able to recover from the misunderstanding when transitioning to the node "Light".

A.2 Dialogues in the slot-based dialogue state representation architecture

This section presents two different example dialogues using the slot-based dialogue state representation architecture. To increase the naturalness of the dialogues, the dialogue acts outputted by the dialogue manager are translated into natural language. The first dialogue is an example of successful dialogue and the second is and example of failed dialogue. Note how the slot based architecture gives more flexibility to the flow of the dialogue, letting the manager to choose the questions needed to collect the necessary information in any order. This might lead to faster dialogues (e.g. If the system is unsure about the slot 2 value, it can ask for the slot 1 value and use this information to disambiguate on the value of the second slot).

Dialogue 4: Goal = *TV-Channel-Two*

User	*Channel* (Speaks the command "Channel")
System	*Ask slot 1* (Asks: *What is the device you want to control?*)
User	*TV* (Speaks the command "TV")
System	*Confirm slot 2* (Asks: *Do you want to change the volume?*)
User	*No* (Speaks the command "No")
System	*Ask slot 2* (Asks: *What functionality do you want to control?*)
User	*Channel* (Speaks the command "Channel")
System	*Confirm slot 2* (Asks: *Do you want to change the channel?*)
User	*Yes* (Speaks the command "Yes")
System	*Ask slot 3* (Asks: *What channel do you want?*)
User	*Two* (Speaks the command "Two")
System	*Confirm goal* (Asks: *Do you want to change the TV to channel two?*)
User	*Yes* (Speaks the command "Yes")
System	*Perform top hypothesis* (Performs action: *TV-Channel-Two*)

As the action performed matches the goal, the dialogue ends as a successful dialogue.

Dialogue 4: Goal = *Light-Off*

User	*Light* (Speaks the command “Light”)
System	*Ask slot 1* (Asks: *What is the device you want to control?*)
User	*Light* (Speaks the command “Light”)
System	*Confirm slot 1* (Asks: *Do you want to control the bluray?*)
User	*No* (Speaks the command “No”)
System	*Ask slot 2* (Asks: *What functionality do you want to control?*)
User	*Off* (Speaks the command “Off”)
System	*Confirm goal* (Asks: *Do you want to turn the TV off?*)
User	*No* (Speaks the command “No”)
System	*Ask slot 1* (Asks: *What device do you want to control?*)
User	*Light* (Speaks the command “Light”)
System	*Perform top hypothesis* (Performs action: *Hifi-off*)

As the action performed does not match the goal, the dialogue ends as a failed dialogue. Note how the system struggles to understand the command “Light” and tries to infer it from the value of the second slot. However, the value of this slot is “Off”, which is a value common to all devices, so it does not help. It also tries to confirm the value of the first slot twice, but finally it chooses the wrong action. Even if the dialogue fails, this dialogue shows the higher flexibility of slot-based architectures compared to tree-based ones.

Appendix B

User simulation

In this appendix, the parameter configurations of the simulated users used in chapters 3, 4 and 5 are explained in more detail. In the first section, the configuration of the behaviour models is explained and in the second section, the configuration of the generative ASR simulation model.

B.1 Behaviour model

The behaviour models (equation 3.1) used vary depending on the dialogue state representation architecture used. In chapters 3 and 4, a behaviour model able to interact with the tree-based architecture is used, and in chapter 5, a behaviour model able to interact with the slot-based architecture.

B.1.1 Tree-based architecture

This section describes the parameter configuration of the user behaviour model used in chapters 3 and 4. When the system uses the tree-based dialogue state representation (see section 3.2.2), the dialogue structure imposed by this representation makes the set of rules to follow by the user behaviour model very simple. Firstly, at the beginning of each dialogue, the simulated user has to choose one goal. In this simulated user configuration, the goal is chosen at random from the set of possible goals of the system (defined by the ontology presented in figure 3.7), following an uniformly random distribution. If it is assumed that the user fully understands the system functionalities (i.e. knows every command he/she has to say to reach the goal in each node of the tree) and does not change his/her goal, the set of rules that the user follows is:

- **Case 1**: The system is in a (correct) tree node waiting for an user command

$\rightarrow$ The user will say the command of the goal corresponding to the tree node it is in.

- **Case 2**: The system asks the user to repeat his last command $\rightarrow$ The user will repeat his last command.
- **Case 3**: The system has transitioned to a wrong tree node and is waiting for an user command $\rightarrow$ The user will say the meta-command "Back".

B.1.2 Slot-based architecture

This section describes the parameter configuration of the user behaviour model used in chapter 5. This behaviour model is more complex than the one described in the previous section, as it needs to interact in a mixed initiative environment, thus has to be able to handle more dialogue situations. In addition, this behaviour model does not assume that the user fully understands the system functionalities, implementing a mechanism to simulate user errors. Different user error rates are implemented to simulate different user expertise levels. In addition, the user goal in each dialogue is not drawn from an uniform distribution, introducing small variations in the probability of choosing each goal.

The set of parameters that control the user behaviour are the following:

- G_{sdev}: Goal distribution standard deviation $\rightarrow$ standard deviation (before normalising) of the probability to chose each goal.
- C_c: Command confusion rate $\rightarrow$ the probability to confuse (change) a command by the user.
- C_{sdev}: Command confusion standard deviation $\rightarrow$ standard deviation of the probability to confuse a command by the user.
- C_l: Command confusion limit $\rightarrow$ the size of the set of commands that can replace the original in case of confusion.

When the behaviour model is instantiated, the goal distributions and command confusion rates are randomly sampled using the set of parameters described above. To sample the goal distribution, the model starts with a uniform multimodal distribution. Then, each category (goal) in this distribution is deviated by an amount sampled by the standard deviation defined by G_{sdev}. Finally, the distribution is renormalised to sum to one.

To create the command confusion rates, a similar approach is taken. The models starts with a uniform multimodal distribution in which the categories are the possible user commands (36 in this case) and each command has a probability C_c to be changed by other. Then, each category (command) in this distribution is deviated by an amount sampled by the standard deviation defined by C_{sdev}. In addition, for each command, a list of commands of length C_l is randomly sampled. Whenever a command is confused, it will be interchanged by a command included in its corresponding list.

To generate the dialogue corpus described in section 5.1, three different parameter configurations are used, simulating three different user expertise levels: *novice*, *intermediate* and *expert*.

- *novice* $\rightarrow G_{sdev} = 0.05$, $C_c = 0.1$, $C_{sdev} = 0.05$ and $C_l = 5$.
- *intermediate* $\rightarrow G_{sdev} = 0.05$, $C_c = 0.05$, $C_{sdev} = 0.05$ and $C_l = 5$.
- *expert* $\rightarrow G_{sdev} = 0.05$, $C_c = 0.0$, $C_{sdev} = 0.0$ and $C_l = 0$.

In each dialogue, the user behaviour model will chose one goal sampled at random from the goal distribution. Then, the set of rules that the user follows in each turn is:

- **Case 1**: The system is waiting for a user action $\rightarrow$ The user will say the value of one of the slots corresponding to the goal. The slot whose value is said is chosen at random.
- **Case 2**: The system asks the user for the value of a slot $\rightarrow$ The user will say the value of that slot.
- **Case 3**: The system asks a confirmation question for the value of a slot $\rightarrow$ The user will answer "yes" or "no".
- **Case 4**: The system asks the user to repeat his last command $\rightarrow$ The user will repeat his last command.
- **Case 5**: The system asks a confirmation question for the whole goal $\rightarrow$ The user will answer "yes" or "no", depending on if the full goal is correct or no.

B.2 Generative ASR simulation model

The generative ASR simulation model used in chapters 3 and 4[1], simulates the noise introduced to the true user action by the ASR channel, converting a user

[1]The experiments in chapter 5 use the sampled ASR simulation model presented in section 3.2.3.

command into an N-best list of commands with associated confidence scores (see section 3.2.3). This model is approximated as:

$$P(\omega|u) = P(\tilde{\mathbf{u}}, \mathbf{c}|u) \simeq \prod_{i=1}^{n} P(\tilde{u}_i|u) \prod_{i=1}^{n} P(c_i|i) \tag{B.1}$$

This equation depends on two different probability distributions: the command confusion model $P(\tilde{u}_i|u)$ and the confidence score model $P(c_i|i)$.

The command confusion model, $P(\tilde{u}_i|u)$, models the probability of a command $\tilde{u}$ appearing in the position i of the N-best list when the true user command is u. A different command confusion model is learnt for each speaker-ASR pair. Therefore, for each (speaker, amount, u, i) tuple, a multinomial distribution has to be learnt, where the classes are the possible values of $\tilde{u}$ (the set of 36 commands). These probabilities are learnt from the statistics[1] obtained in the experiments in section 3.1.2.

To learn the confidence score model, the same approach taken by Williams et al. (2005) is taken, where different probability distributions are learnt depending only on if the N-best hypothesis in that position is correct or not (the confidence scores do not depend on the actual command u). Therefore, $P(c_i|i) = P_{cor}(c_i|i)$ if $\tilde{u}_i = u$ (if the confidence score corresponds to the true user action) and $P(c_i|i) = P_{inc}(c_i|i)$ otherwise. As for the command confusion model, $P_{cor}(c_i|i)$ and $P_{inc}(c_i|i)$ are learnt from the statistics obtained from the experiments in section 3.1.2, estimating the probability density function using a Gaussian kernel density estimator[2].

Finally, the generated N-best list with confidence scores must be post-processed to satisfy the following three constrains:

- Two commands cannot be repeated in the N-best list: $\tilde{u}_i \neq \tilde{u}_j \, \forall \, j \neq i$
- Each confidence score must be less or equal than the confidence scores above it in the N-best list: $c_i \geq c_j \, \forall \, i > j$
- The confidence scores must sum to one: $\sum_{i=1}^{n} c_i \leq 1$

To do so, the command hypotheses $\tilde{u}_i$ are sampled one by one, from $\tilde{u}_1$ to $\tilde{u}_n$. Then, whenever a hypothesis $\tilde{u}_i$ is sampled, if that hypothesis already exists in the N-best list, it is sampled again, until a hypothesis which is not in the N-best list is generated. In the same way, each time a confidence score c_i is sampled, if it is

[1]Only the statistics of half of the UASpeech recordings are used to train this model, as the other half are used to train the POMDP models.

[2]Using the python library Scikit-learn, `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.stats.gaussian_kde.html`.

greater than c_{i-1}, then it is set equal to it, $c_i = c_{i-1}$. Finally, if the confidence scores sum less than one, the remaining probability mass is assigned to the commands not in the N-best list. If the confidence scores sum more than one, the confidence scores are normalised to sum to one.

Bibliography

P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 2004.

M. Asada, E. Uchibe, and K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 1999.

H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 1995.

J. Bayer, C. Osendorfer, D. Korhammer, N. Chen, S. Urban, and P. van der Smagt. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*, 2013.

R. Bellman. Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences*, 1956.

Y. Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2009.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.

D.P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995.

C.M. Bishop. Pattern recognition. *Machine Learning*, 2006.

D. Bohus and A. Rudnicky. A "K hypotheses+ other" belief updating model. In *Proc. of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems*, 2006.

H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, and D. Mostefa. Semantic annotation of the french media dialog corpus. In *Ninth European Conference on Speech Communication and Technology*, 2005.

S.A. Borrie, M.J. McAuliffe, and J.M. Liss. Perceptual learning of dysarthric speech: A review of experimental studies. *Journal of Speech, Language, and Hearing Research*, 2012.

H.A. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, 2012.

J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, et al. The AMI meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, 2005.

I. Casanueva, H. Christensen, T. Hain, and P. Green. Adaptive speech recognition and dialogue management for users with speech disorders. In *Proceedings of INTERSPEECH*, 2014.

I. Casanueva, T. Hain, H. Christensen, R. Marxer, and P. Green. Knowledge transfer between speakers for personalised dialogue management. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2015.

I. Casanueva, T. Hain, and P. Green. Improving generalisation to new speakers in spoken dialogue state tracking. In *Proceedings of INTERSPEECH*, 2016a.

I. Casanueva, T. Hain, M. Nicolao, and P. Green. Using phone features to improve dialogue state tracking generalisation to unseen states. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2016b.

A.R. Cassandra, L.P. Kaelbling, and M.L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, 1994.

S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin. Co-adaptation in spoken dialogue systems. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 2014.

L. Chen, P.H. Su, and M. Gašić. Hyper-parameter Optimisation of Gaussian Process Reinforcement Learning for Statistical Dialogue Management. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2015.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

H. Christensen, S. Cunningham, C. Fox, P. Green, and T. Hain. A comparative study of adaptive, automatic recognition of disordered speech. In *Proceedings of INTERSPEECH*, 2012a.

H. Christensen, S. Siddharth, P. O'Neill, Z. Clarke, S. Judge, S. Cunningham, and M. Hawley. SPECS - An embedded platform, speech-driven environmental control system evaluated in a virtuous circle framework. In *In proc. Workshop on Innovation and Applications in Speech Technology*, 2012b.

H. Christensen, M. Aniol, P. Bell, P. Green, T. Hain, S. King, and P. Swietojanski. Combining in-domain and out-of-domain speech data for automatic recognition of disordered speech. In *Proceedings of INTERSPEECH*, 2013a.

H. Christensen, I. Casanueva, S. Cunningham, P. Green, and T. Hain. homeService: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition. In *Proceedings of the 4th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, 2013b.

H. Christensen, I. Casanueva, S. Cunningham, P. Green, and T. Hain. Automatic selection of speakers for improved acoustic modelling: Recognition of disordered speech with sparse data. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014.

H. Christensen, M. Nicolao, S. Cunningham, S. Deena, P. Green, and T. Hain. Speech-enabled environmental control in an AAL setting for people with speech disorders: a case study. In *Proceedings of the IET International Conference on Technologies for Active and Assisted Living (TechAAL)*, 2015.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

J. Clark and R. Roemer. Voice controlled wheelchair. *Archives of physical medicine and rehabilitation*, 1977.

A. Cohen and D. Graupe. Speech recognition and control system for the severely disabled. *Journal of Biomedical Engineering*, 1980.

M.H. Cohen, J.P. Giangola, and J. Balogh. *Voice user interface design*. Addison-Wesley Professional, 2004.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech & Language*, 2010.

L. Daubigney, M. Gašić, S. Chandramohan, M. Geist, O. Pietquin, and S. Young. Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system. In *Proceedings of INTERSPEECH*, 2011.

L. Daubigney, M. Geist, and O. Pietquin. Off-policy learning in large-scale POMDP-based dialogue systems. In *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 1980.

N. Dehak, P.J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

M.P. Deisenroth, C.E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 2009.

J.R. Deller, D. Hsu, and L.J. Ferrier. On the use of Hidden Markov Modelling for recognition of dysarthric speech. *Computer Methods and Programs in Biomedicine*, 1991.

L. Deng, J. Li, J.T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J.D. Williams, et al. Recent advances in deep learning for speech research at Microsoft. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

T.G. Dietterich. The MAXQ Method for Hierarchical Reinforcement Learning. In *ICML*, 1998.

J.R. Duffy. *Motor speech disorders: Substrates, differential diagnosis, and management*. Elsevier Health Sciences, 2013.

W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 1997.

L. El Asri, R. Laroche, and O. Pietquin. Reward Function Learning for Dialogue Management. In *STAIRS*, 2012.

J.L. Elman. Finding structure in time. *Cognitive science*, 1990.

P. Enderby. Frenchay dysarthria assessment. *British Journal of Disorders of Communication*, 1980.

Y. Engel. *Algorithms and representations for reinforcement learning*. PhD thesis, Hebrew University, 2005.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *ICML*, 2003.

Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 2004.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, 2005.

G. Evermann and P. Woodland. Posterior probability decoding, confidence estimation and system combination. In *Proc. Speech Transcription Workshop*, 2000a.

G. Evermann and P.C. Woodland. Large vocabulary decoding and confidence estimation using word posterior probabilities. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2000b.

M. Fatemi, L.E. Asri, H. Schulz, J. He, and K. Suleman. Policy Networks with Two-Stage Training for Dialogue Systems. *arXiv preprint arXiv:1606.03152*, 2016.

C. Fox, Y. Liu, E. Zwyssig, and T. Hain. The Sheffield wargames corpus. *Proceedings of INTERSPEECH*, 2013.

M. Gales and S. Young. The application of hidden Markov models in speech recognition. *Foundations and trends in signal processing*, 2008.

M.J. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 1998.

M. Gašić, N. Mrkšić, P. Su, D. Vandyke, T.H. Wen, and S. Young. Policy committee for adaptation in multi-domain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

M. Gašić. *Statistical dialogue modelling*. PhD thesis, University of Cambridge, 2011.

M. Gašić and S. Young. Gaussian processes for POMDP-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

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

M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

J.L. Gauvain and C.H. Lee. MAP estimation of continuous density HMM: theory and applications. In *Proceedings of the workshop on Speech and Natural Language*, 1992.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of artificial intelligence research*, 2010.

A. Genevay and R. Laroche. Transfer Learning for User Adaptation in Spoken Dialogue Systems. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016.

K. Georgila, J. Henderson, and O. Lemon. Learning user simulations for information state update dialogue systems. In *Proceedings of INTERSPEECH*, 2005.

K. Georgila, J. Henderson, and O. Lemon. User simulation for spoken dialogue systems: learning and evaluation. In *Proceedings of INTERSPEECH*, 2006.

D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. A form-based dialogue manager for spoken language applications. In *Proceedings of the fourth International Conference on Spoken Language (ICSLP)*, 1996.

A. Graves and N. Jaitly. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *ICML*, 2014.

P. Green, R. Marxer, S. Cunningham, H. Christensen, F. Rudzicz, M. Yancheva, A. Coy, M. Malavasi, and L. Desideri. Remote Speech Technology for Speech Professionals-the CloudCAST Initiative. In *Proceedings of the 6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, 2015.

P. Green, R. Marxer, S. Cunningham, H. Christensen, F. Rudzicz, M. Yancheva, A. Coy, M. Malavasi, L. Desideri, and F. Tamburini. CloudCAST-Remote Speech Technology for Speech Professionals. *Proceedings of INTERSPEECH*, 2016.

D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.N. Chen, J. Gao, L. Deng, and Y.Y. Wang. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. *Proceedings of INTERSPEECH*, 2016.

M. Harper. IARPA Babel Program, 2014.

M.S. Hawley. Speech recognition as an input to electronic assistive technology. *The British Journal Of Occupational Therapy*, 2002.

M.S. Hawley, P. Enderby, P. Green, S. Cunningham, S. Brownsell, J. Carmichael, M. Parker, A. Hatzis, P. O'Neill, and R. Palmer. A speech-controlled environmental control system for people with severe dysarthria. *Medical Engineering & Physics*, 2007a.

M. Hawley, S. Cunningham, F. Cardinaux, A. Coy, S. Seghal, and P. Enderby. Challenges in developing a voice input voice output communication aid for people with severe dysarthria. *Proceedings of the AAATE-Challenges for Assistive Technology*, 2007b.

J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop on knowledge and reasoning in practical dialogue systems*, 2005.

M. Henderson. *Discriminative methods for statistical spoken dialogue systems*. PhD thesis, University of Cambridge, 2015a.

M. Henderson. Machine Learning for Dialog State Tracking: A Review. In *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*, 2015b.

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

M. Henderson, B. Thomson, and S. Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

M. Henderson, B. Thomson, and J.D. Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014a.

M. Henderson, B. Thomson, and J.D. Williams. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014b.

M. Henderson, B. Thomson, and S. Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014c.

M. Henderson, B. Thomson, and S. Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014d.

H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *the Journal of the Acoustical Society of America*, 1990.

H. Hermansky and P. Fousek. Multi-resolution RASTA filtering for TANDEM-based ASR. In *Proceedings of INTERSPEECH*, 2005.

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

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

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

E. Horvitz and T. Paek. A computational architecture for conversation. In *UM99 User Modeling*. Springer, 1999.

H. Jiang. Confidence measures for speech recognition: A survey. *Speech communication*, 2005.

K. Jokinen and M. McTear. Spoken dialogue systems. *Synthesis Lectures on Human Language Technologies*, 2009.

R. Jonson. Dialogue context-based re-ranking of ASR hypotheses. In *2006 IEEE Spoken Language Technology Workshop (SLT)*, 2006.

M.I. Jordan. Serial order: A parallel distributed processing approach. *Advances in psychology*, 1997.

D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

F. Jurčíček, B. Thomson, S. Keizer, F. Mairesse, M. Gašić, K. Yu, and S. Young. Natural belief-critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In *Proceedings of INTERSPEECH*, 2010.

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

F. Jurčíček, B. Thomson, and S. Young. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 2012.

R. Kadlec, M. Vodolán, J. ich Libovickỳ, J. Macek, and J. Kleindienst. Knowledge-based dialog state tracking. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014.

L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

S. Keizer, M. Gašić, F. Jurčíček, F. Mairesse, B. Thomson, K. Yu, and S. Young. Parameter estimation for agenda-based user simulation. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2010.

H. Kim, M. Hasegawa-Johnson, A. Perlman, J. Gunderson, T.S. Huang, K. Watkin, and S. Frame. Dysarthric speech database for universal access research. In *Proceedings of INTERSPEECH*, 2008.

S. Kim and R.E. Banchs. Sequential labeling for tracking dynamic dialog states. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

S. Kim, L.F. D'Haro, R.E. Banchs, J.D. Williams, and M. Henderson. The fourth dialog state tracking challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*, 2016.

A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

S. Larsson and D.R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural language engineering*, 2000.

N.D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 2004.

S. Lee. Structured discriminative model for dialog state tracking. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

S. Lee and M. Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

E.L. Lehmann and J.P. Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

O. Lemon, K. Georgila, and J. Henderson. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *2006 IEEE Spoken Language Technology Workshop (SLT)*, 2006.

E. Levin, R. Pieraccini, and W. Eckert. Using Markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 2000.

W. Li, J. Glass, N. Roy, and S. Teller. Probabilistic dialogue modeling for speech-enabled assistive technology. *Proceedings of the second Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, 2013.

X. Liu, A. Celikyilmaz, and R. Sarikaya. Natural language understanding for partial queries. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

B. Lucas. Voicexml. *Communications of the ACM*, 2000.

F. Mairesse, M. Gašić, F. Jurčíček, S. Keizer, B. Thomson, K. Yu, and S. Young. Spoken language understanding from unaligned data using discriminative classification models. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.

D. Martínez, P. Green, and H. Christensen. Dysarthria intelligibility assessment in a factor analysis total variability space. In *Proceedings of INTERSPEECH*, 2013.

D. Martínez, E. Lleida, P. Green, H. Christensen, A. Ortega, and A. Miguel. Intelligibility Assessment and Speech Recognizer Word Accuracy Rate Prediction for Dysarthric Speakers in a Factor Analysis Subspace. *ACM Transactions on Accessible Computing (TACCESS)*, 2015.

M.J. Matarić. Reinforcement learning in the multi-robot domain. In *Robot colonies*. Springer, 1997.

M.F. McTear. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. *development*, 1998.

X. Menendez-Pidal, J.B. Polikoff, S.M. Peters, J.E. Leonzio, and H.T. Bunnell. The nemours database of dysarthric speech. In *Proceedings of the fourth International Conference on Spoken Language (ICSLP)*, 1996.

H.M. Meng, C. Wai, and R. Pieraccini. The use of belief networks for mixed-initiative dialog modeling. *IEEE Transactions on Speech and Audio Processing*, 2003.

K.T. Mengistu and F. Rudzicz. Comparing humans and automatic speech recognition systems in recognizing dysarthric speech. In *Canadian Conference on Artificial Intelligence*, 2011.

G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of INTERSPEECH*, 2013.

A. Metallinou, D. Bohus, and J.D. Williams. Discriminative state tracking for spoken dialog systems. In *ACL*, 2013.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, 2010.

V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

N. Mrkšić, D.O. Séaghdha, B. Thomson, M. Gašić, P.H. Su, D. Vandyke, T.H. Wen, and S. Young. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*, 2015.

H. Murveit, P. Monaco, V. Digalakis, and J. Butzberger. Techniques to achieve an accurate real-time large-vocabulary speech recognition system. In *Proceedings of the workshop on Human Language Technology*, 1994.

A.Y. Ng, S.J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, 2000.

M. Nicolao, H. Christensen, S. Cunningham, S. Deena, P. Green, and T. Hain. A framework for collecting realistic recordings of dysarthric speech - the homeService corpus. In *The International Conference on Language Resources and Evaluation*, 2016.

S.J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.

J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 2008.

R. Pieraccini and J. Huerta. Where do we go from here? Research and commercial spoken dialog systems. In *Proceedings of the 6th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2005.

R. Pieraccini, D. Suendermann, K. Dayanidhi, and J. Liscombe. Are we there yet? Research in commercial spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, 2009.

O. Pietquin and R. Beaufort. Comparing ASR modeling methods for spoken dialogue simulation and optimal strategy learning. In *Proceedings of INTERSPEECH*, 2005.

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 2006.

O. Pietquin and S. Renals. ASR system modeling for automatic evaluation and optimization of dialogue systems. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 2011.

J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, 2003.

P. Price. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, 1990.

M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

J. Quiñonero-Candela and C.E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 2005.

A. Ragni, K.M. Knill, S.P. Rath, and M.J. Gales. Data augmentation for low resource languages. In *Proceedings of INTERSPEECH*, 2014.

W.J. Rapaport. Logical foundations for belief representation. *Cognitive Science*, 1986.

C.E. Rasmussen. Gaussian processes for machine learning. *Citeseer*, 2006.

C.E. Rasmussen, M. Kuss, et al. Gaussian Processes in Reinforcement Learning. In *NIPS*, 2003.

A. Raux, B. Langner, D. Bohus, A.W. Black, and M. Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Proceedings of INTERSPEECH*, 2005.

H. Ren, W. Xu, and Y. Yan. Markovian discriminative modeling for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

K. Richmond, R. Clark, and S. Fitt. On generating combilex pronunciations via morphological analysis. In *Proceedings of INTERSPEECH*, 2010.

V. Rieser and O. Lemon. Cluster-based user simulations for learning dialogue strategies. In *Proceedings of INTERSPEECH*, 2006.

K. Rosen and S. Yampolsky. Automatic speech recognition and a review of its functioning with dysarthric speech. *Augmentative and Alternative Communication*, 2000.

N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 2000.

F. Rudzicz. Acoustic transformations to improve the intelligibility of dysarthric speech. In *Proceedings of the second Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, 2011.

F. Rudzicz, A.K. Namasivayam, and T. Wolff. The TORGO database of acoustic and articulatory speech from speakers with dysarthria. *Language Resources and Evaluation*, 2012.

J. Schatzmann. *Statistical user and error modelling for spoken dialogue systems*. PhD thesis, University of Cambridge, 2008.

J. Schatzmann and S. Young. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 2009.

J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 2006.

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

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

K. Scheffler and S. Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proc. NAACL Workshop on Adaptation in Dialogue Systems*, 2001.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

B. Schölkopf and A.J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

D. Schuler and A. Namioka. *Participatory design: Principles and practices*. CRC Press, 1993.

M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics 9*, 2003.

H.V. Sharma and M. Hasegawa-Johnson. State-transition interpolation and MAP adaptation for HMM-based dysarthric speech recognition. In *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*, 2010.

H.V. Sharma and M. Hasegawa-Johnson. Acoustic model adaptation using in-domain background models for dysarthric speech recognition. *Computer Speech & Language*, 2013.

D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 2002.

S.P. Singh, M.J. Kearns, D.J. Litman, and M.A. Walker. Reinforcement Learning for Spoken Dialogue Systems. In *NIPS*, 1999.

T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. *arXiv preprint arXiv:1207.1412*, 2012.

N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

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

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

P.H. Su, M. Gašić, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.H. Wen, and S. Young. Continuously Learning Neural Dialogue Management. *arXiv preprint arXiv:1606.02689*, 2016a.

P.H. Su, M. Gašić, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.H. Wen, and S. Young. On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems. *ACL*, 2016b.

S. Sukhbaatar, A. Szlam, G. Synnaeve, S. Chintala, and R. Fergus. Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*, 2015.

K. Sun, L. Chen, S. Zhu, and K. Yu. The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 2014.

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*, 1999.

S. Sutton, D.G. Novick, R. Cole, P. Vermeulen, J. de Villiers, J. Schalkwyk, and M. Fanty. Building 10,000 spoken dialogue systems. In *Proceedings of the Fourth International Conference on Spoken Language (ICSLP)*, 1996.

M.E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 2009.

B. Thomson. *Statistical methods for spoken dialogue management*. Springer Science & Business Media, 2013.

B. Thomson and S. Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 2010.

B. Thomson, K. Yu, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, and S. Young. Evaluating semantic-level confidence scores with multiple hypotheses. In *Proceedings of INTERSPEECH*, 2008.

B. Thomson, M. Gašić, M. Henderson, P. Tsiakoulis, and S. Young. N-Best error simulation for training spoken dialogue systems. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, 2012.

G. Tur and R. De Mori. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

U.N. UN report. World population ageing: 1950-2050. *Department of Economic and Social studies*, 2002.

M. Vacher, D. Istrate, F. Portet, T. Joubert, T. Chevalier, S. Smidtas, B. Meillon, B. Lecouteux, M. Sehili, P. Chahuara, et al. The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2011.

M. Vacher, S. Caffiau, F. Portet, B. Meillon, C. Roux, E. Elias, B. Lecouteux, and P. Chahuara. Evaluation of a context-aware voice interface for ambient assisted living: qualitative user study vs. quantitative system evaluation. *ACM Transactions on Accessible Computing (TACCESS)*, 2015.

P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 2008.

V. Vukotic, C. Raymond, and G. Gravier. Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding? In *Proceedings of INTERSPEECH*, 2015.

M.A. Walker, D.J. Litman, C.A. Kamm, and A. Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, 1997.

Y.Y. Wang, L. Deng, and A. Acero. Spoken language understanding. *IEEE Signal Processing Magazine*, 2005.

Z. Wang and O. Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

W. Ward and S. Issar. Recent improvements in the CMU spoken language understanding system. In *Proceedings of the workshop on Human Language Technology*, 1994.

T.H. Wen, M. Gašić, N. Mrksic, L.M. Rojas-Barahona, P.H. Su, S. Ultes, D. Vandyke, and S. Young. A Network-based End-to-End Trainable Task-oriented Dialogue System. *arXiv preprint arXiv:1604.04562*, 2016.

P.J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1988.

J.D. Williams. Applying POMDPs to dialog systems in the troubleshooting domain. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007a.

J.D. Williams. Using particle filters to track dialogue state. In *2007 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2007b.

J.D. Williams. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *IEEE Journal of Selected Topics in Signal Processing*, 2012.

J.D. Williams. Multi-domain learning and generalization in dialog state tracking. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

J.D. Williams. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

J.D. Williams and S. Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 2007.

J.D. Williams, P. Poupart, and S. Young. Factored partially observable Markov decision processes for dialogue management. In *Proc. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005.

J.D. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

S. Young. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2000.

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

S. Young. CUED standard dialogue acts. *Report, Cambridge University Engineering Department, 14th October*, 2007.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 2010.

S. Young, M. Gašić, B. Thomson, and J.D. Williams. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 2013.

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

V. Zue, S. Seneff, J.R. Glass, J. Polifroni, C. Pao, T.J. Hazen, and L. Hetherington. JUPITER: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 2000.