THE UNIVERSITY *of* EDINBURGH

Generative Probabilistic Models of Goal-Directed Users in Task-Oriented Dialogs

Aciel Eshky

Doctor of Philosophy

Institute for Language, Cognition and Computation

School of Informatics

University of Edinburgh

2014

Abstract

A longstanding objective of human-computer interaction research is to develop better dialog systems for end users. The subset of user modelling research specifically, aims to provide dialog researchers with models of user behaviour to aid with the design and improvement of dialog systems. Where dialog systems are commercially deployed, they are often to be used by a vast number of users, where sub-optimal performance could lead to an immediate financial loss for the service provider, and even user alienation. Thus, there is a strong incentive to make dialog systems as functional as possible immediately, and crucially prior to their release to the public. Models of user behaviour fill this gap, by *simulating* the role of human users in the lab, without the losses associated with sub-optimal system performance. User models can also tremendously aid design decisions, by serving as tools for *exploratory analysis* of real user behaviour, prior to designing dialog software.

User modelling is the central problem of this thesis. We focus on a particular kind of dialogs termed *task-oriented dialogs* (those centred around solving an explicit task) because they represent the frontier of current dialog research and commercial deployment. Users taking part in these dialogs behave according to a set of *user goals*, which specify what they wish to accomplish from the interaction, and tend to exhibit *variability* of behaviour given the same set of goals. Our objective is to capture and reproduce (at the semantic utterance level) the range of behaviour that users exhibit while being consistent with their goals.

We approach the problem as an instance of generative probabilistic modelling, with explicit user goals, and induced entirely from data. We argue that doing so has numerous practical and theoretical benefits over previous approaches to user modelling which have either lacked a model of user goals, or have been not been driven by real dialog data. A principal problem with user modelling development thus far has been the difficulty in evaluation. We demonstrate how treating user models as probabilistic models alleviates some of these problems through the ability to leverage a whole raft of techniques and insights from machine learning for evaluation.

We demonstrate the efficacy of our approach by applying it to two different kinds of task-oriented dialog domains, which exhibit two different sub-problems encountered in real dialog corpora. The first are informational (or slot-filling) domains, specifically those concerning flight and bus route information. In slot-filling domains, user goals take categorical values which allow multiple surface realisations, and are corrupted by

speech recognition errors. We address this issue by adopting a topic model representation of user goals which allows us capture both synonymy and phonetic confusability in a unified model. We first evaluate our model intrinsically using held-out probability and perplexity, and demonstrate substantial gains over an alternative string-goal representations, and over a non-goal-directed model. We then show in an extrinsic evaluation that features derived from our model lead to substantial improvements over strong baseline in the task of discriminating between real dialogs (consistent dialogs) and dialogs comprised of real turns sampled from different dialogs (inconsistent dialogs).

We then move on to a spatial navigational domain in which user goals are spatial trajectories across a landscape. The disparity between the representation of spatial routes as raw pixel coordinates and their grounding as semantic utterances creates an interesting challenge compared to conventional slot-filling domains. We derive a feature-based representation of spatial goals which facilitates reasoning and admits generalisation to new routes not encountered at training time. The probabilistic formulation of our model allows us to capture variability of behaviour given the same underlying goal, a property frequently exhibited by human users in the domain. We first evaluate intrinsically using held-out probability and perplexity, and find a substantial reduction in uncertainty brought by our spatial representation. We further evaluate extrinsically in a human judgement task and find that our model's behaviour does not differ significantly from the behaviour of real users.

We conclude by sketching two novel ideas for future work: the first is to deploy the user models as transition functions for MDP-based dialog managers; the second is to use the models as a means of restricting the search space for optimal policies, by treating optimal behaviour as a subset of the (distributions over) plausible behaviour which we have induced.

Lay Summary

A longstanding objective of human-computer interaction research is to develop better dialog systems for end users. Where these dialog systems are commercially deployed, they are often to be used by a vast number of users, where sub-optimal performance could lead to an immediate financial loss for the service provider, and even user alienation. Thus, there is a strong incentive to make dialog systems as functional as possible immediately, and crucially prior to their release to the public. Models of user behaviour fill this gap, by *simulating* the role of human users in the lab, without the losses associated with sub-optimal system performance. User models can also tremendously aid design decisions, by serving as tools for *exploratory analysis* of real user behaviour, prior to designing dialog software.

User modelling is the central problem of this thesis. We focus on a particular kind of dialogs termed *task-oriented dialogs* (those centred around solving a task) because they represent the frontier of current dialog research and commercial deployment. Users taking part in these dialogs behave according to a set of *user goals*, which specify what they wish to accomplish from the interaction, and tend to exhibit *variability* of behaviour given the same set of goals. Our objective is to capture and reproduce (at the semantic utterance level) the range of behaviour that users exhibit while being consistent with their goals.

We approach the problem as an instance of generative probabilistic modelling, with a model of user goals, and which we induce automatically from data. We argue and demonstrate that doing so has numerous practical and theoretical benefits over previous approaches to user modelling which have either lacked a model of user goals, or have been not been driven by real dialog data. We demonstrate the efficacy of our approach by applying it to two different kinds of task-oriented dialogs, the first are informational (slot-filling) domains, such as flight booking and but route enquiry, and the second are spatial navigation dialogs, which are novel for statistical dialog modelling.

Acknowledgements

I would like to thank my supervisor, Mark Steedman, for the substantial thought and time he has invested in my project, and for constantly challenging my ideas, which has made me a better researcher. I am also grateful to Subramanian Ramamoorthy for valuable input at the later stages of my PhD. Special thanks to Ben Allison for in-depth discussions, many of which have changed the way I approach and solve problems. Thanks to my examiners, James Henderson and Steve Renals for taking time to examine my work, for a pleasant and enjoyable viva, and for valuable comments which have helped improve the final presentation of my work.

Thanks to members of my research group for motivating discussions and occasional whimsy, including: Bharat Ram Ambati, Christos Christodoulopoulos, Emily Thomford, Greg Coppola, Kira Mourão, Mark Granroth-Wilding, Mike Lewis, Prachya (Arm) Boonkwan, Tom Kwiatkowksi, and Tejaswini Deoskar. Thanks to colleagues who have proofread parts of my thesis, including: Annie Louis, Aris Valtazanos, Ben Allison, Bharat Ram Ambati, and Eva Hasler; and to those who have proofread my papers, including: Ali Eslami, Micha Elsner, Mike Lewis, and Sasa Petrovic.

I am deeply appreciative of the financial support I have received from King Saud University as a recipient of a PhD Scholarship. I am also grateful for the additional support I have received from the School of Informatics at the University of Edinburgh.

I owe a great dept of thanks to many friends and family members, too numerous to list here. Special thanks to my husband Ben for skilfully crafting delicious meals to fuel my research, and to my children, Teema and Adam, for good company and welcome distractions.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Aciel Eshky*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern computing is pervasive and ubiquitous. Recent years have witnessed an explosion in the number of devices in circulation, and a change in the nature of devices in use. While twenty years ago the predominant paradigm of computing was a desktop environment, the majority of devices being sold today would not be recognised as computers in a previous era: smartphones, tablets, watches, and even smart-glasses. With these new devices simultaneously evolved changes in interaction: where the keyboard and mouse previously dominated, much interaction is now done with touch-screen devices. More relevant to this thesis, voice-driven interaction has started to appear on these devices: a famous example being Siri, the name given to Apple's iOS personality, to whom you can speak and give instructions, and receive answers as speech. Siri is an example of a simple *dialog system*: a piece of software which engages in conversation with humans. Dialog systems are the broader focus of this thesis.

There are many environments in which the ability to converse with a computer is not just desirable, but essential. In hostile environments, traditional input devices may not be usable. Visually impaired users may not be able to make effective use of traditional peripherals. Older people, or those without the exposure to computing, may feel much more at ease with an established means of interaction (dialog) than being forced to learn an alien means. Many computer systems are accessed remotely by phone, but presently require a human to sit at the other end of the line, and interface with the computer on behalf of the caller, because the breadth of functionality offered by the system cannot yet be reliably exposed over a conversational interface. The last of these items, the phone interaction, has seen some limited success and commercial deployment: systems for travel booking, restaurant information, and meeting scheduling have been deployed commercially (Williams, 2012). However, in the space of all

possible conversations these certainly rank amongst the simplest that could be held: in these dialogs, one of the participants is a computer which is tasked with providing the human user with information in a very specific domain.

What has prevented deployment in more complex domains? In part, although the study of dialog and of automated interlocutors is relatively old, it relies on technologies that have until recently not been sufficiently mature: speech recognisers, semantic parsers, natural language generators, and text-to-speech engines must all be relatively robust for the dialog problem to be manageable. However, another hindrance has been the lack of a solid and accepted framework for modelling the dialog decision process, such that advances in one domain provide at the very least insight (if not direct traction) when applied to another. Each dialog system has been a complex and specific series of rules specifying behaviour exhaustively (Strik et al., 1997; Suendermann et al., 2009). More recently, the notion of automated dialog management as Markov Decision Processes (MDP)[1] has emerged as a standard in which dialogs can be represented, and the correct move inferred (Levin et al., 1998; Roy et al., 2000; Singh et al., 2002; Williams and Young, 2007; Henderson and Lemon, 2008). This has led to some degree of formalisation as to the mechanics of dialogs. With this formalisation comes the ability to apply a whole raft of techniques and insights from machine learning to the construction of dialog systems. Just as with many other areas of Artificial Intelligence in the last thirty or forty years (speech recognition, machine translation, parsing, information retrieval, image annotation, to name a few), this sets the stage for big leaps in the level of functionality we can expect from dialog systems.

This is the context in which this thesis is set: the adoption of a more formal statistical paradigm for human-computer dialogs. While this paradigm has been adopted quite successfully for learning and executing the machine's *policy* in a dialog[2], there is another aspect of the dialog which still largely clings to the older, more explicit, and less general means of determining action: that of modelling the human user's role in the dialog: this is the central problem of this thesis.

Modelling the human user's role in dialog has a particular significance in the Markov Decision Process paradigm, where it acts as the *environment* in which the computer *agent* finds itself. This makes the origins of the MDP as a scheme of robot control clear: where the agent is a robot placed into an environment about which it

[1] We use "Markov Decision Process" to refer broadly to the paradigm, which includes the Partially Observable and Relational variants.

[2] A policy is what dictates how an action is to be chosen for execution by the dialog manager at a particular point in the dialog.

initially knows nothing, and where learning the consequences of its actions becomes vital. In dialog there is no physical environment, but rather the response to the system's actions is complementary user actions, both of which can advance the state of the dialog (ultimately to the point at which *success* occurs). Thus the agent must learn which of its actions can elicit which of the user's actions, and ultimately which sequence of actions lead to a satisfactory resolution of the dialog. However, placing the agent in the company of a real human is infeasible due to the complexity of the dialog space: because of the branching factor of moves available at each time, the number of possible dialogs is astronomical, and even exposing the agent to a reasonable and representative subset of them just once would exhaust the patience of the unlucky human many times over. Thus simulated users are employed, either as the sum total of the training of a dialog system, or at least to initialise its knowledge of the dynamics to a non-degenerate condition. These simulated users are actually just models of user behaviour, however, like the older dialog systems they have been designed to supersede, they are often build with large amounts of manual engineering (Schatzmann et al., 2007b,a; Schatzmann and Young, 2009) and are not portable to new domains, nor do they enable promising and exciting new types of human-machine dialogs.

Outside of the MDP framework, modelling users has many other (although perhaps slightly softer) uses. For instance, when designing a dialog system, some exploratory analysis of user behaviour might reveal that there are certain categories of users with markedly different behaviours, all of whom should be catered for if the system is to achieve widespread success. This kind of exploratory analysis can be described as a usability-motivated approach to user modelling. Further towards the psychological end of the dialog research spectrum, building models of users allows the testing of linguistically-motivated hypotheses about the way in which people conduct themselves when speaking to one another, how their dialog strategies evolve over time or in different situations, or indeed what the fundamental principles of communication are.

Throughout this thesis, we will refer to two schools of thought in dialog research to which this thesis's approach to user modelling is relevant: an *engineering* mindset, focused on the construction of effective dialog systems, with little regard to the plausibility of the mechanism which underpins them beyond its efficacy. The MDP use for user models as user simulators has fallen squarely into this category: the model is a means to a definitive end, namely a more successful dialog system. We also refer to what we term a *behavioural* view of dialog research, which encompasses both linguistically-motivated theories, and exploratory usability-motivated research,

in which understanding the motives and processes underpinning the advancement of conversation is of paramount importance. While it may at first seem that the goals of these two strands of research are at best orthogonal, in fact we will demonstrate that some behavioural insights and a focus on more plausible user behaviour can lead to improvements in objective downstream applications, most significantly in tasks of direct relevance to the construction of dialog systems.

1.1 The problem

We have until this point been using the term 'dialog' simply to refer to an interaction between a human and computer in the medium of speech. Here we clarify that for the purposes of this thesis, we address a specific subset of dialog interactions, which goes some way beyond what has been previously attempted in the literature. Primarily, we are concerned with *co-operative*, *task-oriented dialogs* between two interlocutors. As their name suggests, these dialogs are centred around a task, and the purpose of the dialog is to collaboratively solve the task.

Task-oriented dialogs have been the subject of much research in the literature, both on the engineering and the behavioural strands of research. While most natural dialogs do not fall into this category, most of the applications of automated dialog systems thus far (and certainly those within the statistical MDP paradigm) have been of the task-oriented variety. These dialogs are the simplest because their purpose is clear, and it is often easy to tell when they have concluded to the participants' satisfaction. Examples include informational (or slot-filling) dialogs, such as flight booking and bus route information provision. In this thesis we also go beyond these to a different domain: a spatial navigation domain, which adds an interesting bi-modal aspect to the communication, namely spatial reasoning in conjunction with conversational modelling. Arguably all these domains are quite simple in the space of all possible conversations with users, especially when contrasted with open-ended dialogs, but their solutions are still important because they simultaneously represent the frontier of current research and the most readily solvable class of problems for which there is genuine use in the current technological and computational environment.

As we have stated in the previous section, we are interested in modelling the role of human interlocutors, as opposed to determining a dialog manager's optimal decision process. More specifically, we model the behaviour of users as *utterances*, but at the

more abstract level of a semantic representation[3], rather than directly from acoustics or natural language. Semantic representations allow us to abstract away from many of the specifics of language and focus on the intended meaning (in other words, we model 'what to say' and not 'how to say it').

Key to our modelling are two concepts which relate to the behaviour users tend to exhibit in task-oriented dialogs. The first is **goal-directedness**: users behave according to a set of goals specifying what they wish to accomplish from the dialog. While the set of goals is not usually observed as part of the dialog, it still largely the behaviour of users. The concept of goal-directedness is also known as 'consistency' in the literature and will be key to our modelling. The second concept is concern the **variability** that users tend to exhibit while being consistent with their goals. Our aim to capture the a range of behaviour that users exhibit given the same underlying goals.

As we have motivated in the previous section, this kind of modelling has direct applications for the construction of dialog systems. We advocate the use of models which can be used as tools for exploratory analysis of interlocutor behaviour, which necessitates inducing the models from data instead of hand-building them. At the same time, we advocate the use of models which can be deployed as tools for user simulation with which an MDP-based dialog system interacts to learn optimal policies of behaviour. This necessitates the ability to generate behaviour from our model (and not only predict behaviour), and the ability to define complete distributions of behaviour with a probability indicating the behaviour's plausibility.

To summarise we require:

1. a model that captures, and is able to generate, the range of goal-directed behaviour that humans exhibit (modelling)

2. a means of inducing the model entirely from data, to enable us to use them as tools for exploratory analysis (learning)

3. a means of inspecting or validating how accurately the model captures user behaviour (evaluation)

[3]Modelling at the semantic level is in line with previous approaches to user modelling in the literature. It has also been referred to in the literature as the 'intention' or 'concept' level.

1.2 Approach

In this thesis we propose an empirical, data-driven approach to modelling users in co-operative task-oriented dialogs. More specifically, we treat user modelling as an instance of generative probabilistic modelling, with an explicit model of user goals and induced entirely from data without prior access to the domain's vocabulary. Our approach is based on existing statistical models and techniques, but represents the first attempt to apply these methods to model user behaviour, at the semantic utterance level, in task-oriented dialogs.

Modelling an explicit user goal allows us to capture goal-directed user behaviour. Adopting a data-driven approach alleviates much of the manual effort typically involved in the construction of a user model for the purpose of simulation (for MDP-based dialog systems). Furthermore, user models based on empirical observations of human behaviour can help uncover unexpected behaviour, which makes them viable for use as tools for exploratory analysis. Additionally, an empirical approach places fewer assumptions about user behaviour than handcrafted alternatives, and removes the need for speculation.

Probabilistic modelling allows us to express notions of variability and uncertainty in a mathematically principled way. They also allow us to separate the problems of model design, model induction, and inference: a separation which allows for a range of machine learning algorithms for learning and inference to be deployed regardless of the model structure. It also makes the models largely domain-independent and easily extensible. The use of generative probabilistic models in particular offers us further advantages. Generative models define not only the probability of observing behaviour, but of generating it, thus we can use them as simulators by sampling from the distributions they provide. They are also designed to provide complete distributions over behaviour, with a probability proportional to the behaviour's plausibility, making them suitable for capturing variability of user behaviour as an array ranging from the highly plausible to the highly implausible. Finally, the distributions learnt can be inspected for a better understanding of user behaviour.

The ability for generative models to both assign a probability to behaviour and to generate behaviour offers new avenues for evaluation. Like other probabilistic models, they are able to assign probability to held-out test-sets, as an indicator of how probable a model finds unseen data enabling the comparison of the performance of models relative to one another, but additionally, using the model in generation mode allows us

to assess the quality of its output.

1.3 Contributions

The contribution of this thesis is in demonstrating that generative probabilistic models provide a flexible and mathematically principled framework for capturing and reproducing goal-directed user behaviour in task-oriented dialog domains. We show how generative models of user behaviour can be defined with an explicit model of user goals. We show how they can be induced automatically from standard dialog resources without prior access to the domain's vocabulary. We further show how to evaluate them intrinsically and extrinsically. More specifically, in the course of this thesis, we demonstrate the following:

- Adopting a generative probabilistic approach allows us to capture the range of behaviour that users exhibit.
- Modelling an explicit user goal allows us to better capture user behaviour compared to only modelling observed behaviour.
- The models can grow to accommodate the complexity of real world problems (we demonstrate this by applying our modelling framework to two problems in Chapters 5 and 6).
- Models of goal-directed user behaviour of sufficient complexity can be automatically induced from data without handcrafting, and without prior access to the domain's vocabulary.
- The models can generalise beyond observed training instances to deal with unseen instances, and produce novel forms.
- The models can be assessed and formally evaluated in isolation from dialog managers.

We conclude this thesis by proposing new ways of using the models which we induce to arrive at some of the objectives we have stated in this chapter, namely: to arrive at optimal dialog management policies. In the final chapter, we sketch two new ideas for future work of obtaining optimal behaviour. The first is to deploy the user models as the transition function for an MDP-based dialog manager. The second is to

consider optimal behaviour of a conversational agent as a subset of the distribution of plausible behaviour which we have induced.

1.4 Thesis outline

Chapter 2: Background In this chapter, we motivate the problem of user modelling as an important step towards building better dialog systems. We describe the kinds of domains which are of interest, namely, cooperative task-oriented dialog domains, and describe the kind of behaviour users exhibit within these domains, with an emphasis of goal-directedness and variability. We relate the problem of user modelling to the problem of dialog management. We then present the problem of dialog management as a Markov decision process, and discuss the role user modelling traditionally plays within the MDP framework.

Chapter 3: Models of Dialog Users in the Literature In this chapter we review the user modelling literature as a continuum, which at one extreme constructs systems driven by procedures and explicit rules to produce behaviour that resembles that of users, while at the other extreme defines formal probabilistic models which treat user behaviour as observations arising from a stochastic process. We then describe the three classes of evaluation approaches in the literature, namely: evaluating the predictions of the user models, evaluating synthetic dialogs resulting from the interaction of user models with a dialog manager, and finally, evaluating the dialog management policy learnt by interacting with the user model, and we highlight the strengths and weaknesses of each of the approaches.

Chapter 4: Goal-Directed Users as a Data-Driven Generative Model In this chapter, we introduce our own approach to modelling and evaluation, which builds and expands upon some of the ideas presented in the previous chapter. In our modelling framework user behaviour is treated as an instance of generative probabilistic modelling, with an explicit model of user goals to capture goal-directed behaviour, and which we induce entirely from data without handcrafting or prior access to the domain's vocabulary. We highlight the strengths of this approach compared to previous approaches, which include, generality, adaptability to new domains, learnability from data, and its ability to capture not only observable dialog behaviour but also latent structure. Finally, we motivate evaluating the models in isolation from a dialog manager, and present the standard machine

learning evaluation metrics and ideas adopted in this thesis. The modelling framework presented in this chapter is used to develop models in the remainder of the thesis.

Chapter 5: Modelling Users with Latent Categorical Goals In this chapter, we apply the modelling framework to two standard dialog domains of the slot-filling variety conceding travel information. Key to ensuring consistency of user behaviour in slot-filling domains is the notion of user goals. User goals in these domains take categorical values which allow multiple surface realisations (e.g., New York, NYC). Because the data for these domains is typically collected over noisy channels, surface realisations are further corrupted by speech recognition errors (e.g. London, Oakland). We describe the most commonly used representation of user-goals in the slot-filling literature: a string-based representation, and demonstrate its limitations in producing plausible user behaviour. We then present our own novel approach which adopts a topic model representation of user goals in conjunction with a bigram model to capture simple conversational dynamics. Our topic model representation captures both synonymy and phonetic confusability in a unified model, which we demonstrate is better able generate plausible user behaviour. We apply our modelling to two standard dialog resources: the DARPA Communicator (flight information) and Let's Go (bus information). We evaluate the models intrinsically using held-out probability and perplexity, and demonstrate substantial gains over the alternative string-based goal representation of goals, and over a (non-goal-directed) pure bigram model. To further demonstrate the strength of our model, we evaluate it in a downstream task. We train an SVM classifier over features derived from our model to discriminate between real dialogs (consistent dialogs) and dialogs comprised of real turns sampled from different dialogs (inconsistent dialogs). We show that features derived from our model lead to substantial improvement over strong baselines in performing this task. The work presented in this chapter was published in Eshky et al. (2012).

Chapter 6: Modelling Users with Spatial Goals This chapter describes the application of the modelling framework to a novel dialog domain where the task is spatial navigation. The novelty in this task lies in the form which user goals take: instead of taking a categorical form, user goals are spatial trajectories across a landscape, represented in raw data as pixel coordinates. The dispar-

ity between the the spatial routes and their grounding in utterance creates an interesting challenge. We demonstrate how to abstract from spatial routes to a meaningful representation, using an efficient feature based representation which we derive. This feature-based representation facilitates reasoning and admits generalisation to new routes which have not been previously encountered by the model. The probabilistic representation allows us to capture a range of plausible behaviour given the same underlying goal, a property which is exhibited by human users in the domain. We apply our modelling to the HCRC Map Task corpus. We evaluate our model intrinsically using held-out probability and perplexity, and find a substantial reduction in uncertainty brought by our spatial representation. We evaluate our spatial component extrinsically, first using accuracy, and then qualitatively in a human judgement task. We find that despite a low percentage of match between our model's behaviour and that of humans, our model's behaviour is not judged to be significantly different from the behaviour of real users. The work presented in this chapter was published in Eshky et al. (2014).

Chapter 7: Conclusion and Future Work To conclude this thesis, we return to the objectives set out in the introduction and review how the approach taken in this thesis has been useful in addressing them. We summarise the specific results achieved in the previous chapters, and present several avenues for future research.

Chapter 2

Background

The aim of this chapter is to provide appropriate background for much of the discussion, as well as novel work, that follows. We begin by introducing the relevant definitions, and then present some of the background useful for understanding where the contributions of this thesis sit.

2.1 Applications of user modelling

In this thesis, we take the first steps to bridging the gap between *behavioural dialog research*, aimed at understanding human behaviour in dialog, and *engineering dialog research*, aimed at automating the process of constructing better dialog systems for end users. We view these objectives as complementary, in that better behavioural insights can lead to gains in downstream tasks.

One application of user modelling is to deploy them as tools for exploratory analysis of the behaviour of users prior to constructing dialog systems. This process can undercover unexpected user behaviour and can thus tremendously aid and inform the design of dialog systems. This application can be classed as behavioural in the sense that it helps shed light on human behaviour in dialog, but also as engineering because it can lead into insights for building better dialog systems. A second application of user modelling, which is perhaps more direct, is for the automatic optimisation of dialog managers, and in particular those formulated as a Markov Decision Process. User modelling has particular significance in the formalism as it plays the role of the environment in which the dialog manager (or MDP-based agent) finds itself. In Section 2.3 we get into more detail on how user models can serve either as an external, 'sample model' for the MDP to interact with and learn optimal policies of behaviour, or alter-

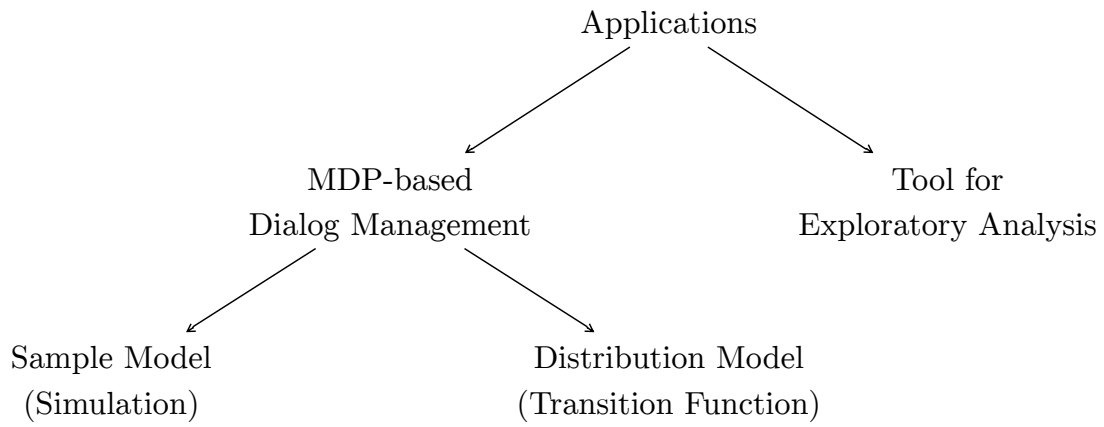


Figure 2.1: A diagram illustrating the motivation behind developing user models as adopted in this thesis. 'Leaf nodes' in this diagram represent possible applications of user models. User models can aid dialog system design by serving as tools for exploratory analysis of real user behaviour, prior to designing dialog software. More directly, user models can be exploited by MDP-based dialog systems, which learn optimal behaviour through interaction.

natively, as an explicit 'distribution model' or 'transition function internal to the MDP. Although the second of these options has been relatively under-explored in the dialog literature, it gives us access to a different class of reinforcement learning algorithms to be deployed (namely: 'exact algorithms', or 'planning methods'). Figure 2.1 illustrates some of the applications of user modelling.

2.2 Definitions

2.2.1 Task-oriented dialogs

This thesis focuses on *task-oriented* dialogs: as their name suggests, these dialogs are centred around a task, and the purpose of the dialog is to complete it. Examples of tasks include modem troubleshooting (Boye, 2007; Williams, 2007), spatial navigation (Thompson et al., 1993; Janarthanam et al., 2013), and travel information provision (Levin et al., 2000a; Parent and Eskenazi, 2010). These kinds of dialogs can be contrasted with open-ended dialogs in which there is no task. In fact, this is more of a spectrum than a binary separation, as certain dialogs exhibit properties of both (a doctor consulting with their patients, a tour guide showing people around a museum, etc.). Despite being arguably simpler than open-ended dialogs, much of

the research in dialog has been dedicated to task-oriented dialogs, partly because they have a clear measure of success, with respect to how well the task was accomplished. In contrast, open-ended dialogs are far less constrained, and it is more difficult to judge whether they have concluded successfully (and thus harder to measure whether scientific progress has been made when addressing them). Task-oriented dialogs also most directly translate to commercial applications, making them an exciting and rapidly expanding area of research.

2.2.2 Cooperative interlocutors

Some task-oriented dialogs, certainly those of interest in this thesis, take place between two *cooperative* interlocutors who work together towards completing the task, although, other kinds of task-oriented dialogs, which go beyond the scope of this thesis, can involve more than two interlocutors, such as robotic bartending (Giuliani et al., 2013), or can be adversarial in nature, such as strategic games (Asher et al., 2012).

In certain tasks, one of the interlocutors can be considered an expert while the other a novice, as is the case in troubleshooting domains, however, this is not a necessary distinction for task-oriented dialogs. In fact, one of the tasks we address in this thesis, namely spatial navigation, assumes that the two cooperative interlocutors take different roles simply as a result of having access to different kinds of information.

2.2.3 Goal-directed users

One notion which is central to this thesis is that of *user goals*. User in task-oriented dialogs exhibit what is referred to as goal-directed behaviour, in that they behave in accordance with some goal, or a set of user goals, describing what they wish to accomplish from the interaction. Consider for example flight information provision as a task: the set of goals which belongs to a user aiming to book a flight would include values for their city of origin, their destination, a flight date, etc. Users taking part in a dialog for the purpose of booking a flight would behave in accordance with such goals, in that they would provide information consistent with, and not contradictory to, these user goals. Since this thesis concerns modelling goal-directed user behaviour, user goals will be key to our modelling.

2.2.4 Variability in user behaviour

Another important factor to consider when modelling user behaviour is what we refer to as *variability*, which concerns the wide range of behaviour that users can display. Under identical circumstances, and even for the same set of goals, users can behave very differently, not just to one another but also compared to previous actions they themselves have taken. When modelling how human users behave in these domains, our objective becomes the ability to capture and reproduce the range of behaviour that they can exhibit while still being consistent with their goals.

Note that our objective when modelling user behaviour is not to necessarily learn high quality actions; if we examine the range of behaviour exhibited by users we find that it varies in quality. Some actions taken by users may seem reasonable to us while others may strike us as erroneous or sub-optimal. Neither is the objective to confirm our intuition about user behaviour: some actions taken by users may be ones we expect, while others can highly contradict our intuitions about user behaviour. Instead, the objective is to capture the range of behaviour empirically exhibited by users regardless of how we might perceive its quality. We refer to the range of behaviour that users can exhibit as *plausible behaviour*.

In direct contrast to modelling the range of behaviour that users exhibit is the problem of *control*, where the objective is to determine *optimal behaviour*; in other words, the selection of the best possible action at any given point in the dialog. Control is the function of the decision making component in a dialog system, commonly referred to as the dialog manager. When determining the dialog manager's behaviour the objective is to select optimal action with respect to successfully completing the task or allowing users to achieve their goals[1]. Table 2.1 constant dialog management with user modelling.

Although at a first glance optimal behaviour and plausible behaviour may seen orthogonal, they are in fact complementary. As it will become more apparent throughout this chapter, obtaining one is in fact a prerequisite to accomplishing the other: in order to discover optimal dialog management decisions, a model of plausible user behaviour is not only desirable, but required. The intuition is that in order to determine the best kinds of actions, one needs to know how users tend to respond to these actions, and make decisions accordingly.

[1] Although one can imagine there being a small set of optimal actions, current state-of-the-art dialog managers assume a single optimal action at any given point.

User Model	Dialog Manager
Capturing plausibile actions	Determining optimal actions
A distribution over plausible behaviour	A policy of optimal behaviour
Directed by user goals	Optimised with respect to task objective

Table 2.1: A table contrasting user modelling with automated dialog management. Models of user behaviour capture the range of goal-directed behaviour that users exhibit. In constant, the aim of dialog management is not to exhibit plausible human-like behaviour, but to take the best action at any given point in the dialog. Automated dialog management is typically optimised with respect to a 'task objective'.

2.3 The problem of dialog management

A dialog system consists of various components. A speech recogniser receives acoustic stimulus from a user and transcribes it into a sequence of words, which are then interpreted into a semantic representation using a natural language understanding component (with an underlying semantic parser). Semantic representations allow us to abstract away from some of the specifics of natural language and focus on the intended meaning of the utterance. The dialog manager is designed to operate at this abstract semantic level, and needs to decide what to say, or what semantic utterances to produce, in response to the user stimulus for the purpose of solving the underlying task. On the output generation side, the reverse process is followed: the natural language generation component translates the semantic utterance into a sequence of words which are then synthesised into an acoustic utterance using a speech synthesis component[2]. Figure 2.2 visually depicts these concepts. Much research has been dedicated to each of the five components described above, but the dialog manager is the component relevant to the problem we wish to address.

In the simplest case, a dialog manager would follow a handcrafted dialog strategy, or *policy*, dictating how to behave for every possible user stimulus[3]. Simple handcrafted policies may be appropriate for very small domains, with only a handful of states and actions, especially in the absence of uncertainty or non-determinism. However, as soon as uncertainty or non-determinism are introduced to the problem, these

[2]One variation on this process is to converse directly at the textual (or natural language) level using a keyboard as the user input device.

[3]We use the terms 'strategy' and 'policy' synonymously to refer to the dialog manager's choice of action at a given point in the dialog.

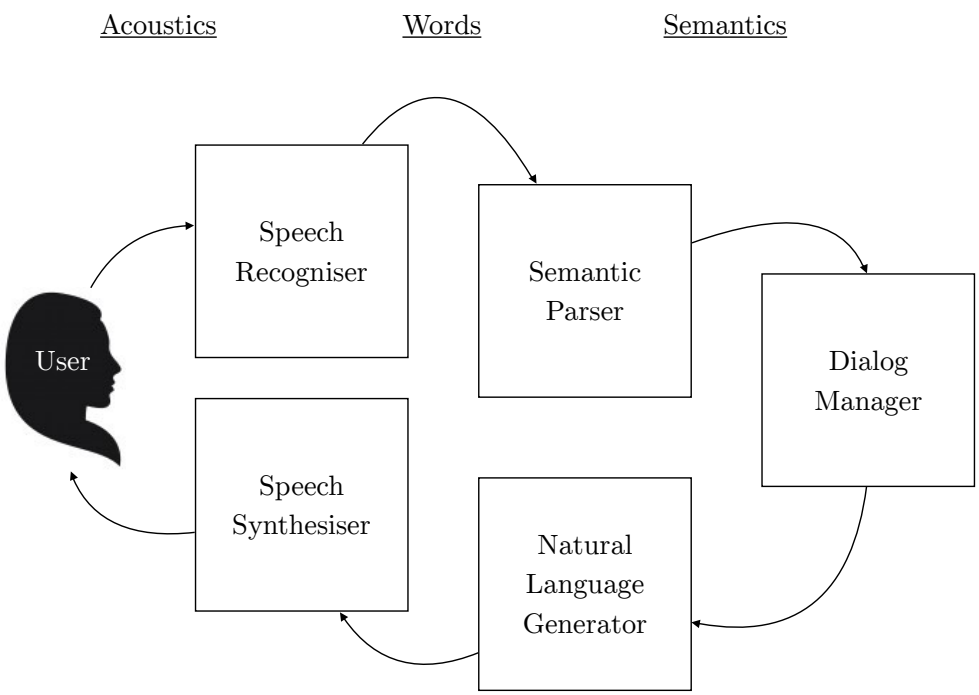


Figure 2.2: A dialog system consists of five components. A speech recogniser processes the acoustic utterance of the user into a sequence of words. The natural language understanding component parses the sequence of words into the appropriate semantic representation. The dialog manager receives as input the semantic utterances of the user, and then needs to decide what to say, or what semantic utterances to produce, in response. The output is then converted into a sequence of words using the natural language generation component, and finally synthesised into an acoustic utterance using the speech synthesis component.

simple strategies fail to cope. Furthermore, because they are handcrafted for a particular domain, it is difficult, or in most cases impossible, to transfer them to new domains, forcing dialog system engineers to laboriously handcraft policies for every possible new domain.

The alternative to handcrafting policies is to automate the process of discovering them by allowing the dialog manager to select actions automatically, a concept which can be referred to as automated control or automated dialog management. If the dialog manager bases its decision of what to say only on the latest observed user stimulus, then the resulting policy would be appropriate locally, but would ignore long-range consequences of the behaviour. Thus for the dialog manager to make more coherent

decisions, future effects of its actions must be incorporated into its decision process. An appropriate framework, which explicitly accounts for long-range effects of actions, is the Markov decision process (MDP) (Sutton and Barto, 1998). The basic idea of MDP is to treat each decision, not in isolation given only present information, but as one of a sequence of decisions made to optimise a long-range task objective. In the last two decades, the Markov decision process has become the most prevalent approach for dialog management in the dialog research community[4]. A description of the main concepts of formalism follows.

2.3.1 Dialog management as an MDP

An MDP-based *agent* is tasked with solving a sequential decision problem, in other words, taking a sequence of actions towards solving the task at hand. The agent interacts with an *environment* from which it receives stimulus, and we define the environment to be anything external to the agent. The agent typically keeps track of an environment state, which in the simplest case is the latest environment stimulus, but usually contains more information about the configuration of the environment thus far and as perceived by the agent. The agent then needs to decide what action is most optimal at the present state of the environment. Optimal actions are those that maximise a long-term objective defined in terms of the task to be solved. Solving the MDP amounts to discovering the optimal policy of behaviour, or the optimal action to take for every possible state.

The task objective is expressed as numerical quantities assigned to states to signal their desirability from the perspective of the task at hand. There are two kinds of such numerical quantities: local *rewards* and global *values*, both of which are relative quantities indicating the utility of states compared to other states (with no upper or lower bound).

Rewards indicate the utility of states in an immediate sense, while values indicate the utility of states in the long run. Rewards are usually computed as a function of task completion and possibly additional desirable out comes (commercial objectives, such as dialog length, cost of performing certain actions, and user satisfaction). Reward functions are required at the time of formulating the MDP and must therefore be derived in advance prior to solving the MDP. Values, on the other hand, are com-

[4]Dialog management has also been treated as a Partially Observable Markov Decision Process, (POMDP, pronounced *pom-dee-pee*) which is an extension of MDPs that retains uncertainty about the state as a probability distribution called a ‘belief state’.

puted during the process of solving the MDP. More specifically, a value of a state is computed as the total amount of reward the agent can expect to accumulate over time from that state. Where rewards determine the immediate desirability of a state, values indicate long-term desirability by taking into account the states that are likely to follow, and the rewards available in those states. For example, a state might yield a small immediate reward, but has a high value because it is cumulatively followed by other states that yield high rewards, and the inverse can also be true: a state which yields a large immediate reward might have a low value because it is cumulatively followed by other states that yield small rewards. Ultimately, action choices of an MDP-based agent, in other words, policies, are made based on value judgements; the agent seeks actions which bring highest values not highest rewards, because the agent is interested in actions which are optimal globally and in the long run.

Formally, an MDP consists of:

1. S, a set of states
2. A, a set of actions
3. $t : S,A \rightarrow S$, a transition function taking a state and action and returning a new state
4. $r_a(s,s') \rightarrow \mathbb{R}$ a real-valued reward associated with moving from state s to state s', achieved by taking action a

Dialog management can be formulated as an MDP. In this formalism, the concept of an MDP agent corresponds the dialog manager. The actions that the agent can make (the set A) are dialog utterances. The environment in this problem is essentially the user, and states (elements from S) would typically include previous user actions along with variables indicating long-term dialog state (whether certain information had been provided, number of times something had been requested, etc). The transition function is a (possibly probabilistic) mapping from one state to another under a particular action a. Much of what is known about the environment is encoded in a transition fucntion. The reward indicates how good the evolution of state from s to s' is in an immediate sense. In the next sections we turn to the business of maximising reward over the sequence of future actions, which is the essential problem in an MDP and one to which we devote much of the rest of this chapter. For now, Table 2.2 lists some MDP concepts and what they correspond to in dialog, and Figure 2.3 visually illustrates the same ideas.

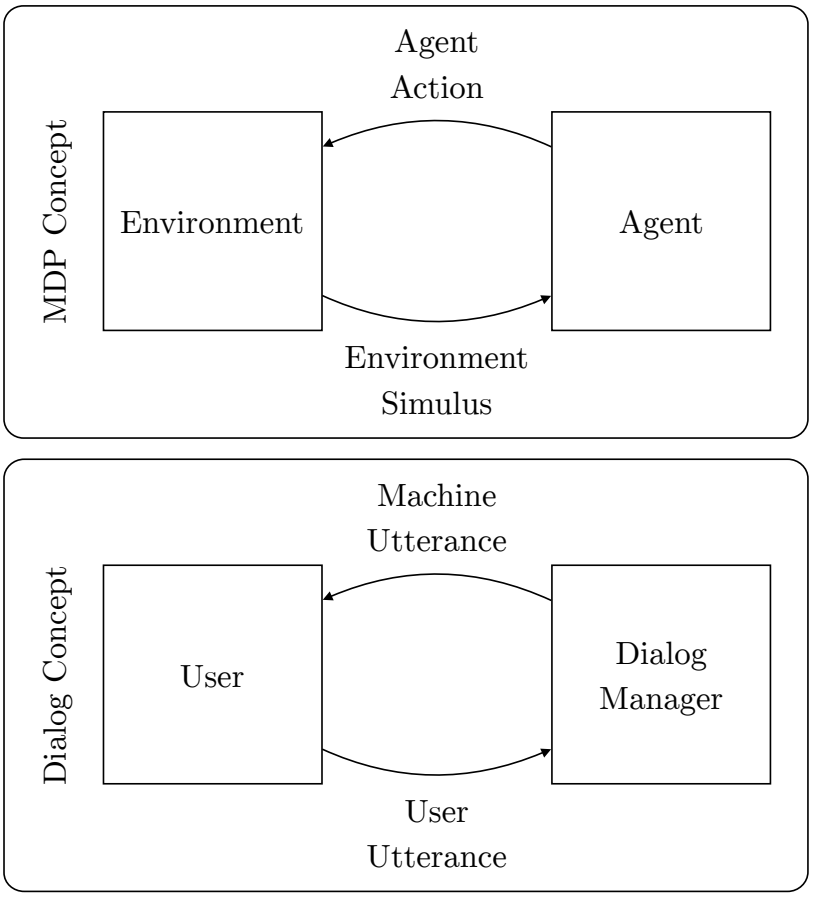


Figure 2.3: An MDP-based agent interacts with an environment from which it receives stimulus, and in response to the stimulus, performs actions. When formulated as an MDP, the dialog manager corresponds to the MDP agent, who interacts, not with a physical environment, but with a user, and produces machine utterances in response to user utterances.

2.3.2 Policies and optimal decisions

A policy can be viewed as a function $\pi : S \rightarrow A$, which tells the manager which action a to take in state s. The goal of learning a dialog manager, or solving an MDP, is to derive the policy π. Typically the goal is to find an optimal policy.

To motivate the notion of optimality, consider that choosing the action a that maximises the reward $r_a(s,s')$ is simple if we have access to the function t. We simply choose the action a which results in the state s' where the reward $r_a(s,s')$ is greatest. If t returns a probability distribution over successor states $p(s'|s,a)$ then this can be straightforwardly extended to choosing the action a that maximises the expected

MDP Concept	Corresponding Dialog Concept
Agent	Dialog Manager
Agent Action	Machine Utterance
Environment	User
Environment Stimulus	User Utterance

Table 2.2: Dialog management can be formulated as a Markov decision process (MDP). In this formalism, the concept of an MDP agent corresponds to the dialog manager, whose actions are machine utterances. The concept of the environment with which the agent interacts (and from which it receives stimulus) corresponds in dialog to the user, who produces user utterances.

reward:

$$\text{Expected reward}(s,a) = \tau(s,a) = \sum_{s'} p(s'|s,a) r_a(s,s') \tag{2.1}$$

However, the problem of dialogs (and in fact all MDPs, or sequential decision problems) is deciding what to do now to maximise total reward in the future. We can formalise this in the current framework by seeking to maximise the following:

$$\psi(s,a) = \sum_{t=0}^{\infty} \max_{a_t} \tau(s_t, a_t) \tag{2.2}$$

Which is to say, we wish to choose the action a now which maximises our total reward in the future (assuming $t = 0$ is now). We cannot approach this problem greedily (i.e. by just picking the action with the largest $\tau(s,a)$) because the rewards we obtain in the future for future actions a_t depend on the state at that time, which depend on the actions we choose now through the transition function. Note also how the summation over τs is to infinity: while many problems will have terminal states, nothing in the paradigm requires some of the states in S to be goal or terminal states. For this reason, it is common to define a *horizon* (a maximum t over which ψ is defined), and also a discount factor $0 < \gamma <= 1$ which allows rewards for low t to be weighted differently to rewards in the distant future (each $\tau(s_t, a_t)$ in (2.2) is multiplied by γ^t). This practically reduces the infinite summation to a summation over a finite horizon, or one that can be suitably well approximated by one.

At its simplest, the policy π is a lookup table which maps a state to the correct action (for dialog, the semantic utterance to be made by the dialog manager when in that state), which is to say the action which maximises ψ. By tracing the set of

utterances to be produced by the manager in response to the set of user stimuli, then the result is effectively a dialog plan; a sequence of actions to be taken from any state, potentially to a terminal state if one has been specified for the problem.

Early MDP-based dialog managers relied on the tabular policies we just described (Kaelbling et al., 1996), which are simple and effective for problems with a small number of states. Most real-world problems, however, do not lend themselves to an enumerable state space, but instead, have unmanageably large, continuous, or even infinite state spaces. In such cases a tabular representation of policies is not possible. In these cases, an alternative is to employ a functional approximation to $\psi(s,a)$ (or of the *value* of the state s, $V(s)$ where $V(s)$ can be seen as :

$$V(s) = \max_a \psi(s,a)$$

Functional policies operate by mapping states to feature vectors, and then treating the approximation of ψ (or V) as a regression problem. Once the regressor is trained, this solution is almost as fast and tractable as the tabular policy but crucially generalises to any state/action pair for which a feature vector can be derived. For these reasons, functional policies have in the last decade replaced tabular policies as the state of the art approach in dialog management (Henderson et al., 2005; Denecke et al., 2005; Lemon et al., 2006; Henderson et al., 2008; Rieser and Lemon, 2008; Thomson et al., 2008; Li et al., 2009; Pietquin et al., 2011a,b; Daubigney et al., 2012).

2.3.3 Models of user behaviour

Now that the framework of dialog management as an MDP has been established, we turn to describing the role which our work would fill within this formalism. In previous sections we covered the notion of a transition function, which is the means by which a state s_t evolves to $s_t + 1$ after action a. In the simplest case, this is a black box which produces a successor state for a given state-action pair. However, if the entire transition function is known as at the time of inducing the policy π, it serves as a model of the environment. As such it can be used to predict how the environment will respond to its actions (Sutton and Barto, 1998). This can be a very powerful concept and opens the door to a different class of learning algorithms. In the domain of dialog, the environment is the user and various attributes of the conversation with the user.

In the reinforcement learning literature, models (transition functions) that produce a description of all possibilities and their probabilities are called **distribution models**,

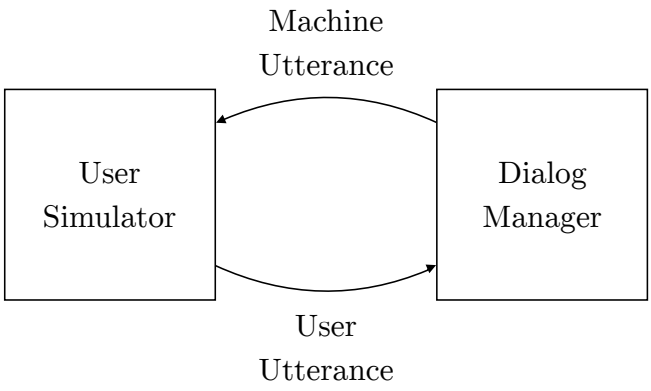


Figure 2.4: An MDP-based dialog manager is able to learn the optimal policy of behaviour through interaction with real users using iterative learning (or reinforcement learning) algorithms. However, because employing real users for this task is expensive and time consuming, user simulators play the role of real users. In reinforcement learning terminology, the user simulator constitutes a *sample model* because it produces a sample of a user utterance in response to a machine utterance (rather than an entire distribution over possible responses).

while those that produce one of the possibilities, sampled according to the probabilities are called **sample models**. Distribution models are more powerful than sample models in that they can always be used to produce samples. However, in many applications, it is much easier to obtain samples than from the transition function than the complete distribution over possibilities. For example, an agent that is expected to learn to interact with its physical environment can simply be placed within the environment, where each sensing action the robot makes results in a sample. In dialog, while iterating with real users is a possibility (Gašić et al., 2011), it has been argued that it is expensive and labour intensive to do so (Eckert et al., 1997).

An alternative to learning by interaction with a physical environment or with real users is to use models to simulate experience. Given an agent action, a sample model produces a possible transition, while a distribution model provides a distribution over all possible transitions. Given a policy, a sample model could produce an entire episode; and a distribution model could produce the distribution over all possible episodes. In either case, we say the model is used to produce simulated experience. Thus far simulators in dialog management have been confined to the sample model variety, and the probabilistic nature of transitions has been somewhat under-explored. We argue that in estimating a distribution model we begin to pave the way for more accurate estimation of values, and thus better policies.

User Model	**Sample Model**	**Distribution Model**
Function	User Simulator	Transition Function
Dialog Management Policy Discovery	Planning Methods (Exact)	Learning Methods (Approximate)

Table 2.3: If a complete model of user behaviour is known ahead of time in the form of a *distribution model*, then it can serve as a transition function. Planning methods, or exact solutions, can then be deployed for process of discovering the optimal dialog management policy. On the other hand, if only samples of user behaviour can be obtained, then the user model is treated as a user simulator. The dialog management policy can then be discovered through interaction using learning methods, or approximate solutions. Note that distribution models can be used as sample models, but that the inverse is not true.

To be concrete: for each action that can be taken by the agent, a **user model** produces a description of the possible user responses and their probabilities. In other words, the user model produces a distribution over user behaviour given the agent's behaviour. When computing probabilities, the user model can also take into account the history of interaction between the agent and the user, or more abstract dialog state monitoring variables. This thesis focuses only on the user response, but the methods deployed are sufficiently general to be further extended to include more than the response. Given the probability of the possible outcomes, and the utility of the outcomes, as computed by the value function, the agent can discover an optimal policy of behaviour, which is the subject of section 2.3.4.

Much of the work in statistical dialog management has addressed the problem of control, or dialog management. In contrast, the problem of user modelling has been the subject of fewer studies and has seen fewer scientific advances. The presentation of this chapter is in part to highlight that better user modelling, and understanding its place in the scheme of dialog management as a whole, has the potential to improve dialog managers too.

2.3.4 Algorithms for optimal policy discovery

So far we have defined what an MDP is, and what makes a policy optimal: an optimal policy picks the a which maximises the expected sum of future rewards, which we

labelled $\psi(s,a)$, for all $s \in S$. As we noted, a related quantity is $V(s)$, the value function for the state s, which is simply $\psi(s,a)$ for the best action a at s. However, we have said nothing about how to go about discovering this policy, which amounts to obtaining the values for ψ or V.

At first it may seem odd to talk about determining the values of these functions as a difficult step. It should be a simple case of computing the relevant summations (as in Equation (2.2)) for all possible s,a pairs. However, one must remember that the cardinality of S, the number of possible states, may be very large or even infinite. This is often the case when a state represents a cross-product of several categorical variables, each of which is of modest dimensionality. For example, if S contains the last user utterance (one of twenty possible, a fairly small number), the grounding status for each of two slots (three possible values each, *unprovided*, *provided* and *grounded*), and the value for each of those two slots (one of a thousand possible entries each, tiny if these are flight destinations, bus stops, etc.), then there are 180 million possible states. These are modest numbers for a dialog. Add to this a set of ten possible dialog manager actions, and there are close to two billion values to compute, each of which requires summing over the set of successor states until a goal state is reached.

To approach solving this problem, there are two fundamental classes of algorithms for MDP policy discovery. At the heart of both is the computation of values as global estimates of the utility of state-action pairs. More specifically, both classes of algorithms are based on looking ahead to future events, computing backed-up values, and using them to update estimates of value functions (Sutton and Barto, 1998), on which policies of behaviour are finally based. There are, however, some fundamental differences between the two classes of algorithms.

The first groups of algorithms require access to a complete transition function, i.e. a complete distribution model of user behaviour (or environment, in MDP terminology). These methods are classed as exact, or **planning methods**. Dynamic Programming and brute force search (exhaustively expand the transition function and search the entire state space for the optimal policy), are two kinds of planning methods for MDP policy discovery. Examples of planning methods for obtaining policies of MDP-based dialog managers include Walker et al. (1998b,a); Young (2000) on exact Dynamic Programming, and Chandramohan et al. (2010) on approximate Dynamic Programming. These methods use algorithms to efficiently but exactly compute the summation in Equation (2.2).

The second class of algorithms are **learning methods**, which do not necessarily

rely on access to a complete and accurate transition function, but are instead able to bootstrap from samples or experience. Another way to think of these is as Monte Carlo estimators to the expectation in Equation (2.2). These treat the model as a black box, capable of generating samples or "experience" of how the environment behaves. Learning methods are then able to estimate a policy directly from raw experience. This is particularly useful in situations where acquiring an accurate distribution model of the environment is hard, but generating experience is easy, for example, when the agent is interacting with a natural system (the physical environment, for instance). However, these methods are also useful because they can learn an optimal policy from an incomplete model of the environment given sufficient samples. To reiterate, learning methods:

1. are compatible with situations where no model, or only a partial model, of the environment can be defined

2. do not require full distributions of environmental responses and can bootstrap from samples

both of which are powerful concepts. Learning methods have been the primary approach for dialog management policy discovery in the literature (Levin et al., 1998; Roy et al., 2000; Singh et al., 2002; Williams and Young, 2007; Henderson and Lemon, 2008; Young et al., 2010).

Young (2000) show examples of optimal dialog policy discovery for both planning and learning methods, but express a preference for the latter. Crucially, for both classes of methods, a model of the environment must exist, either in a distribution form to admit planning, or as an experience-generating black box (at the very least) to admit learning. Table 2.3 illustrates some of these ideas.

2.3.5 A final note

It is customary in dialog management research to first obtain a model of user behaviour (either through handcrafting or by automatically inducing it) and then use it to induce an optimal policy for the dialog manager. Figure 2.5 illustrates this approach as a "two-stage modelling" framework. We follow the same convention, but concern ourselves with the first stage of this process: learning an accurate model of user behaviour. We defer the second stage to future work.

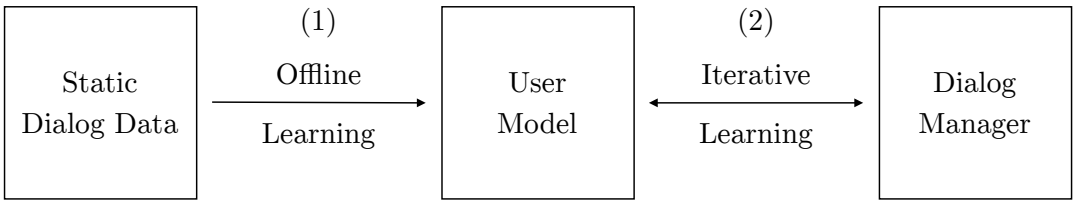


Figure 2.5: It is customary in dialog management research to first obtain a model of user behaviour (either from data or by handcrafting it) and then to use it to induce an optimal policy for the dialog manager. We follow the same convention, but concern ourselves with the first stage of this process: learning an accurate model of user behaviour from static dialog data.

As we have described in Section 2.3.3, there are two kinds of user models: sample models and distribution models. The former are suitable for learning algorithms, while the latter are suitable for planning methods. The models we induce in this thesis can serve as either of those, because we learn complete distributions from which we can sample behaviour.

2.4 Conclusion

The aim of this chapter was to provide appropriate background for much of the discussion, as well as novel work, which will follow. We motivated the problem of user modelling as an important step towards building better dialog systems, either as tools for exploratory analysis of user behaviour, or more directly as sample models / distribution models to be used by MDP-based dialog management.

We started by introducing some of the relevant definitions in Section 2.2. We described the kinds of domains which are of interest, the kind of behaviour users exhibit within these domains, and how modelling user behaviour relates to the wider problem of dialog management.

To help understand where the contribution of this thesis sit, in Section 2.3 we presented some of the background on dialog management as a Markov decision process, and discussed the role user modelling traditionally plays within this framework, namely as a user simulator (or sample model), as well as the role which it can play (a distribution model) if the approach advocated in this thesis is adopted.

Chapter 3

Models of Dialog Users in the Literature

User modelling research aims at providing dialog researchers with models of real user behaviour in order to aid with the design and improvement of dialog systems. Where these dialog systems are commercially deployed, they are often used by vast numbers of users, where sub-optimal system performance could lead to an immediate financial loss for the service provider, or even user alienation and refusal to re-use the system once improved. Thus, there is a strong incentive to make dialog systems as functional as possible immediately, and crucially prior to their release to the public, even when the systems are designed to continually improve their performance through exposure to the behaviour of real users. Models of users fill this role by *simulating* the role of users in the labs, without the risks and losses associated with sub-optimal system behaviour when interacting with real users.

In the literature, user models have been primarily developed as tools for simulation. User simulators are used to test the performance of dialog systems prior to their release to the public. They have also been extensively used to provide interactive training examples for MDP-based dialog systems, which learn optimal policies of behaviour through interaction. In addition to simulation, this thesis advocates the use of models which can be deployed for exploratory analysis of dialog corpora, in order to discover the range of behaviour users can exhibit, and to better cater for their requirements. Figure 3.1 illustrates the interactive process between a dialog manager and a user simulator.

Previous approaches to user modelling, primarily for the purpose of simulation, have fallen somewhere along a continuum, which at one extreme constructs systems

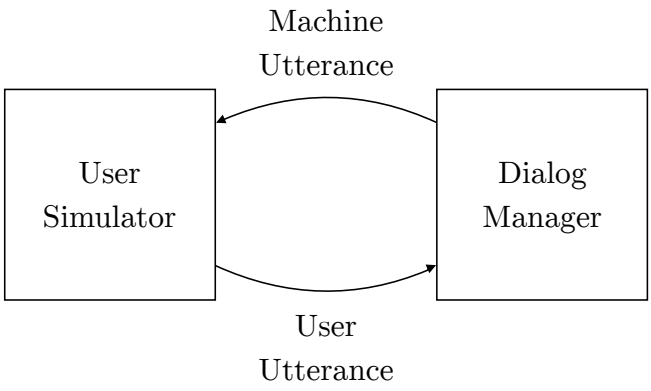


Figure 3.1: User simulators are one kind of user models built for the purpose of testing dialog managers prior to the release of the dialog system to the public. User simulators are also extensively used to train MDP-based dialog managers to discover optimal policies of behaviour through interaction.

driven by procedures and explicit rules to produce behaviour that resembles that of users, while at the other extreme defines formal probabilistic models which treat user behaviour as observations arising from a stochastic process. The work presented in this thesis aims to sit firmly at the modelling rather than procedural end of the spectrum. However, we first provide an overview of the approaches in the field, with particular attention paid to this stochastic/procedural spectrum. Specifically, we concentrate on the notion that when formulating a user simulator as a handcrafted system, it is often possible to quickly incorporate better intuitions of behaviour, but they do not provide any insight into data that can be collected of real user behaviour, and are thus not usable as tools for exploratory analysis. Their assumptions are difficult to test and verify, and thus it can be hard to know whether the intuition that drove their development is supported by real data. In contrast, simulators as models of user behaviour (models in the formal probabilistic sense) require greater initial investment to encode intuition into a formal probabilistic model structure, but they have greater potential to generalise to new data, are transferable to new domains, and their hypotheses can be easily verified and tested using standard evaluation techniques.

3.1 Preliminaries and notation

Modelling user behaviour takes place at the abstract level of a semantic representation (also known as the concept or intention level) and not at the level of natural language or acoustics, because the semantic representation allows us to abstract away from some

of the specifics of language and focus instead on the intended meaning.

Slot-filling, informational domains have been the subject of most dialog management and user simulation research. The task of slot-filling is to accurately elicit from the user (or customer) the values for a set of slots in the presence of ambiguity and automatic speech recognition (ASR) errors. The values are then placed in a query which is submitted to a back-end database, and the results are then returned to the customer. Thus, the dialog manager (which is the decision making component of the dialog system) is not required to reason beyond eliciting accurate values. Although simple in their construction, the difficulty in these domains is as a result of the ambiguity and ASR errors often heavily present in the user stimulus.

Flight booking is an example of a slot-filling domain, where the dialog manager is required to elicit a set of values describing the customer's city of origin (or airport), destination, date, etc. For example, a user might wish to travel from London to New York City on the 17th of July, 2014. The set of **user goals**, g, in this instance are typically represented in the following form:

$$\{\textit{orig\_city}=\textbf{New York City}, \textit{dest\_city}=\textbf{London}, \textit{depart\_date}=\textbf{17-06-14}\}$$

where *orig_city*, *dest_city*, and, *depart_date* are examples of **slots**, s, while London, New York City, and 17-06-14 are examples of **values**, v. The user might wish to convey their origin in an utterance, which at the semantic level can take the following form:

$$\text{PROVIDE } \textit{dest\_city}=\textbf{London}$$

where PROVIDE is an example of a **dialog act**, a. Other dialog acts include CLARIFY, REQUEST, CHECK, and CONFIRM, and they tend to vary from one domain to the other. We refer to utterances at the semantic level as *semantic utterances*, and in particular those belonging to the user as **user utterances** u, and those belonging to the dialog system (or 'machine') as **machine utterances** m. The semantic utterances are what we refer to as **observable behaviour**, and constitute the building blocks of the dialog.

3.2 Stochastic models for user simulation

The first group of user models are the stochastic models, and by stochastic we mean those with a probabilistic formulation. Stochastic models can be divided into two groups: those whose parameters are either set manually or even partially hand-tuned, instead of being driven by real dialog data. We refer to these as *hand-tuned models*

and describe them in Section 3.2.1. The second group of stochastic models are ones that induced fully from data. We refer to these as *data-driven models* and describe in Section 3.2.2. Stochastic models include the following:

1. Eckert et al. (1997) define the Bigram model.

2. Levin et al. (2000b) "restricted-Eckert model": restricts the set of parameters to be induced at the expense of making it domain specific.

3. Pietquin (2004) and Pietquin and Dutoit (2006) goal-directed model encoded in a Bayes Net (BN), but hand-crafted.

4. Georgila et al. (2005a) and Georgila et al. (2006) induce Eckert's model from data, and condition on a longer history (higher order N-gram models).

5. Jung et al. (2009) conditional random fields (CRF).

Out of the six models, 3 is the only model which explicitly models user goals, while 4 and 5 are the only ones fully induced from data.

3.2.1 Hand-tuned models

3.2.1.1 Eckert's model

Stochastic modelling of users in dialog at the semantic level was pioneered by Eckert et al. (1997) for the purpose of debugging and testing dialog systems prior to their release to the public. Eckert et al. introduced a model based on N-grams over semantic utterances: i.e. the semantic user utterance is conditioned on preceding semantic utterances belonging either to the machine or to the user, and modelled these conditional distributions directly. Thus a user's response to a machine utterance is modelled by the conditional probability:

$$p_{ij} = p(U_t = I_i | M_{t-1} = I_j) \tag{3.1}$$

of replying with a response I_i when receiving the stimulus I_j, where I_i and I_j denote sets or sequences of semantic utterances. The process is assumed to be time invariant, i.e. the probabilities do not depend on the absolute value of t but on the sequence of occurrence.

This model has the advantage of being purely probabilistic and fully domain-independent, but does not place enough constraints to capture realistic user behaviour.

Specifically, the generated responses may correspond well to the previous machine action in a local context, but they often do not make sense in the wider context of the dialog. Thus the authors suggest that wider dialog contexts can be incorporated by conditioning on a longer history, as follows:

$$p_{ijkl...} = p(U_t = I_i | M_{t-1} = I_j, U_{t-2} = I_k, M_{t-3} = I_l, ...) \tag{3.2}$$

but restrict the model to a history of 1 (due to data sparsity) and condition a user utterance exclusively on the preceding machine utterance. This model has since been known as the **Bigram model**. Eckert et al. do not train their Bigram model on real data or evaluate the quality of its output, and instead handcraft the conditional distributions based on characteristics inspired by their corpus and on intuition.

3.2.1.2 Levin's model

Levin et al. (2000b) describe how Eckert's model can be altered to limit the number of model parameters and to account for some degree of conventional structure in dialogues. Instead of allowing complete freedom of user responses to machine stimulus, only the probabilities for anticipated types of user responses are calculated for each type of machine behaviour. They formulate a generative model of users using the following set of parameters:

1. Response to greeting: which includes the probability $p(n)$, where n is the number of slots provided by the user in a single utterance; the probability of the slot $p(s)$ (e.g., origin, destination, ...), and the probability of the value of the slot $p(v|s)$ (e.g., "London"|origin)

2. Response to constraining question: the probability of the user specifying a value for some slot (for example, slot s_R) when the system requests the value for a different slot (for example, slot s_G, where R and G simply indicate different slots) and the probability of providing additional unsolicited slots in the same response

3. Response to a relaxation prompt: Parametrised by $p(yes|s_G) = 1 - p(no|k_G)$, which is the probability of accepting or rejecting the proposed relaxation of slot

Because the model is no longer domain or vocabulary independent, it has a limited capacity to be induced from data. For example, the model makes assumptions about dialog initiative (machine-initiative or mixed-initiave, but not user-initiative), and the

kinds of dialog acts to expect (constraining questions and relaxation prompts), both of which are not present in their corpus of choice: the ATIS corpus. Thus, they estimate from data only the first of the parameters of their model and are forced to handcraft the rest.

Besides being almost fully handcrafted, and being domain and vocabulary dependent, another drawback is that the model does not include as part of the generative model a mechanism for generating the values for the slots and thus the model is only partially probabilistic in its formulation.

3.2.1.3 Pietquin's model

The two probabilistic approaches presented so far ensure that the user model provides a locally sensible response to a machine utterance, however, they do not ensure that the responses provided throughout the dialog are consistent with one another. Pietquin (2004) (and Pietquin and Dutoit (2006)) mitigates against this problem by explicitly modelling user goals on which user behaviour can be conditioned, in a similar fashion to the goals presented in Section 3.1. The user goals are defined as a set of slot-value pairs describing the user's desired outcome, and was first introduced by the procedural approach of Scheffler and Young (2002) presented in Section 3.3. Pietquin then defines a set of variables relevant to user modelling and encodes the dependencies between the variables in a Bayesian Network. Note however, that a Bayesian Network simply defines the dependencies between variables in the model, but tells us nothing about the model structure.

Like Eckert's model this approach is purely probabilistic, and domain and vocabulary independent. However, the probabilities in this model are all hand-selected and restricted using intuition, similarly to Levin's model. Pietquin notes that inducing the model from data would require either user goal annotations, or inferring user goals from their observed behaviour, and would additionally require smoothing, all of which are deemed too difficult.

A final note worth mentioning is the recent estimation of Pietquin's model, not from real data, but from *simulated* data (Rossignol et al., 2011), an exercise which is in some sense closer to handcrafting distributions than to inducing them from real data. Rossignol et al. generate simulated data through the interaction of another simulator with some dialog manager. Their own simulator, which is based on Pietquin's model, is then induced from this artificial data. Although they emphasise their ability to estimate the model from data which in part is (deliberately) incomplete, their experimental setup

is highly artificial and is still fairly preliminary, as their data was produced with this experiment in mind.

3.2.2 Data-driven models

Only a small subset of stochastic user models have been induced automatically from data. Georgila et al. (2005a) and Georgila et al. (2006) hypothesise that user utterances conditioned on a longer dialog history would result in more consistent behaviour. They present a fully probabilistic model based on N-grams (similar to that which was first conceived, but manualy set, by eckert97). They induce the model entirely from data, modelling only observable behaviour, without an explicit model of user goals.

Jung et al. (2009) is another approach to inducing user models as Conditional Random Fields (CRF) from data. However, they require vast amounts of annotations (which, for example, include what information has been provided so far in the dialog). As is the case with Georgila et al.'s work, this approach still lacks an explicit model of user goals and assumes that the longer history of contexts gives rise to somewhat consistent user behaviour.

3.3 Procedural approaches for user simulation

An alternative approach to treating user behaviour as arising from a stochastic process, is to construct systems that behave as users would, for the sole purpose of simulation, or in other words, for providing training and testing examples for dialog systems. Because they are not models in the formal sense, we refer to systems designed to mimic user behaviour as procedural approaches, in reference to the underlying procedure governing their behaviour, which is typically a set of rules interleaved with trainable parameters to allow the simulation of variability in behaviour. As is the case with stochastic models, most of the prominent procedural approaches have addressed slot-filling domains, and thus the work presented here is confined to this kind of domains.

One of the most prominent procedural approaches for simulation in the literature is the agenda-based user simulation (Schatzmann et al., 2007b,a; Schatzmann and Young, 2009). Agenda-based user simulation defines a probability distribution over user goals, which can be induced from data or manually specified when no data is available. At simulation times, user goals are generated at the start of the dialog, and an agenda,

which is a stack-like structure of utterances to be produced for each user goal, is then devised using a deterministic procedure.

Prior to agenda-based user simulation, Scheffler and Young (2002) defined a different procedure for simulating behaviour, in which they handcraft rules for actions that depend on user goals, and define probabilistic parameters for the selection of actions that are goal-independent. Prior to the start of a dialog, they map out an extensive decision network which determines user actions for every possibility. Keizer et al. (2010) later combined the procedural decision network with agenda based simulation to allow for more variability in the simulated behaviour.

Another approach by Chung (2004) defines a different handcrafted procedure to govern user behaviour, but allows the simulation of unique dialogs by defining four trainable parameters, which include the probability of terminating a dialog, or the probability of taking initiative of the system.

Procedural approaches can be appealing, because it is often possible to quickly incorporate better intuitions of behaviour as a set of rules, but they require some degree of handcrafting or manual specification of user behaviour, and this has many disadvantages. Firstly, because they are designed to work with specific domains, procedural approaches are of limited transferability to new domains, which would require us to laboriously construct a new simulation procedure for every new domain (or class of domains) we advance to. For example, it is unclear how to transfer procedural approaches designed for slot-filling domains to a spatial navigation domain. Secondly, procedural approaches force us to speculate about user behaviour and place hard assumptions about it, instead of allowing us to observe it empirically. Because their behaviour does not arise from empirical observations, procedural approaches provide no insight into user behaviour, cannot be used for exploratory analysis of dialog corpora, and have no utility beyond simulation. A third issue concerns the limited range of behaviour these approach are designed to exhibit. Although they are constructed for the purpose of simulation, procedural approaches are not often designed with variability in mind. Thus, they display a restricted range of user behaviour, which in turn restricts the access of dialog managers to plausible, and potentially important, states of user behaviour. A final drawback of procedural approaches has to do with evaluation: because they lack a formal probabilistic interpretation they can be difficult to assess how closely they exhibit real user behaviour without recourse to a dialog manager (with which they can interact to simulate behaviour), an approach of evaluation we argue against in Section 3.5.

3.4 Corpora expansion approaches

Another idea related to simulation is that of corpora expansion: the production of static synthetic human-machine dialogs for the purpose of obtaining more data to train and test dialog systems. The most prominent examples of corpora expansion techniques include the work of Cuayahuitl et al. (2005) and that of Li et al. (2013) in slot-filling domains, in which entire dialogs, containing complete user and system interactions, are produced. Both approaches use a combination of Hidden Markov Models (HMM) to model individual subdialogs and a bigram model to model sequences of subdialogs. Like most stochastic approaches, they model only observed behaviour and lack an explicit model of user goals (what they refer to as goals are subsequences of system turns within the same topic, in other words, *subdialogs*). They further assume that there are no ASR errors and no multiple surface realisations of the same value.

Although the flavour of the models they present resembles that of the stochastic approaches, in that they are fully probabilistic, the primary objective of corpora expansion differs to that of simulation. Corpora expansion is motivated by a perceived scarcity of dialog resources, and thus strive to generate synthetic complete interactions that resemble the scarce real interactions. In contrast, user simulation is designed to be interactive, producing relevant examples for dialog systems for iterative learning processes. User modelling presented in this thesis more closely falls under the class of stochastic models of users, and although our models can be used interactively with some dialog system to produce synthetic data (should one wish to do so) this is not the purpose of our research.

3.5 Evaluation of user models

The problem of evaluation is central to the ability to perform research: whether progress is being achieved necessarily assumes some valid and useful way of measuring progress. While for many tasks this is trivial (is the answer correct or not?) for dialog in general and user simulation in particular, it is far from straightforward. In this section we explore the different ways of evaluating a simulator, and the implications these strategies have on the efficacy of the simulator, its quality, and any potential impact on systems which rely on the simulator.

There are no strongly established metrics of evaluation for user simulation. This thesis argues that better models are ones that are better able to capture user behaviour

(i.e., predict and reproduce). A recent review divides approaches based on the level of dialog they asses, namely whether they compute turn-level statistics or dialog-level statistics (Pietquin and Hastie, 2013). We take the following alternative view : evaluation approaches in the literature have fallen under three kinds: those that evaluate the predictive power of the model, those that evaluate synthetic dialogs generated using the model in conjunction with a dialog manager, and finally, those that evaluate the policy learned by a dialog manager trained on the model. Our division of evaluation approaches closely resembles that of Georgila et al. (2005a), and to some extent Schatzmann et al. (2006).

3.5.1 Evaluating model predictions

The first group of metrics evaluate how closely the predictions of a user model resemble that of real user behaviour, exhibited in data not used to train the models.

3.5.1.1 Accuracy, precision and recall

Accuracy, and the closely related metrics of precision and recall, are some of the most widely used metrics for evaluating user simulators in the literature (Schatzmann et al., 2005; Georgila et al., 2006; Rossignol et al., 2011). These metrics measure, at the utterance level, how closely a simulator-predicted utterance resembles that of real humans in the same context. Imagine that decisions are made at an utterance level: given some context in the dialog, there is a simulator-predicted utterance u_s and the utterance the human produced in that context u_h. Given a set of such utterances and their contexts, U, the accuracy of a simulator can be defined as:

$$\text{Accuracy}(U,s) = \frac{\sum_{i=1}^{n} 1_{u_{si}=u_{hi}}}{n} \tag{3.3}$$

Where 1_x is the indicator function which is 1 if the condition x is true, and 0 otherwise. This is simply to say, accuracy is the fraction of utterances where the simulator predicted utterance matches the human utterance.

Closely related are Recall and Precision, except that these are defined for a particular human utterance (type) u_t:

$$\text{Recall}(U,s,t) = \frac{\sum_{i=1}^{n} 1_{u_{hi}=u_t \wedge u_{si}=u_{hi}}}{\sum_{i=1}^{n} 1_{u_{hi}=u_t}} \tag{3.4}$$

$$\text{Precision}(U,s,t) = \frac{\sum_{i=1}^{n} 1_{u_{hi}=u_t \wedge u_{si}=u_{hi}}}{\sum_{i=1}^{n} 1_{u_{si}=u_t}} \tag{3.5}$$

Note that the difference between recall and precision is in the denominator: Recall is the fraction of utterances that should have been u_t that were correctly identified; Precision is the fraction of utterances identified as u_t which truly are. Recall and precision can be averaged over utterance types t to provide global recall and precision for the system; they can also be combined by taking their harmonic mean, producing F-measure.

The issue with accuracy, precision, and recall, is that they all assume that given some context of the dialog there is a single correct response (which we labelled u_{hi}) and a range of incorrect responses; that is to say, the human produced utterance for some context is the only possible or correct response in that context. For many problems this is the case; there is a single ground truth value against which a system is evaluated; however, this is rarely the case for dialog. One of the central points of this thesis is that users tend to exhibit a range of plausible behaviour for the same context, and that the notion of there being a single correct response for a given context is incorrect. Thus, accuracy, precision and recall heavily penalise models exhibiting the valid variability they were intended to capture, and in fact bias us towards worse models that exhibit a smaller degree of variability (a similar concern was also rained in Pietquin and Hastie (2013)). In this thesis, we argue (and in one instance demonstrate) that these metrics are unsuitable for evaluating user models and simulators.

3.5.1.2 Perplexity and held out probability

If the user simulator is a probabilistic model (which is to say, for some context the simulator defines a probability distribution $p_s(u_s)$, then in addition to the metrics of the previous section, we can use evaluations designed for probability models. These have the benefit that they acknowledge that realisations of u in particular contexts are not the only possible realisations, but rather samples from some distribution over what could have occurred in that context. In particular, the held-out probability metric is defined over the same set of held out utterances U as:

$$\text{Held-out probability(U, s)} = \frac{\sum_{i=1}^{n} p_s(u_i)}{n} \tag{3.6}$$

which is to say the mean probability of the utterances u_i under the model of the simulator s.[1] This is a standard way of evaluating a probability model, and uses the intuition

[1] In reality, the probability of the held-out data is the product rather than sum of these probabilities under the assumption that elements in U are sampled i.i.d. conditioned on appropriate context. However, since we often wish to compare simulator performance across sets U of different cardinality, it makes

that if the induced model $p_s(\cdot)$ is good, then the probability assigned to actual observations will be high.

Note however that the target here is not 1: this would again assume that in a particular context some utterance is the only conceivable choice, which is the point against which we have been arguing. Indeed this can often mean that metrics related to held-out probability have little meaning in absolute terms. Rather their strength is in assessing the relative merits of different approaches (although one can make a reasonable measurement of the absolute efficacy of a method by looking at its performance relative to a suitable baseline).

As a final note, in language modelling (an area where probability models are widely employed for similar purposes) it is more common to use perplexity than held out probability, and indeed some user simulation work has used this metric. Perplexity is a monotonic function of held-out probability; however, it is on an exponential scale, and arguably has a slightly more intuitive definition. That is to say, the perplexity can be thought of as roughly the number of possible utterances to be chosen between at any given time. In any case, we will often use the two interchangeably, since if $p_s(u_1) > p_s(u_2)$ then their perplexities has the same ordinal property.

3.5.2 Evaluating synthetic interactions

While the metrics of the previous section, in particular perplexity and held-out probability, have many attractive properties, not all simulators can assign probabilities to utterances (especially procedural simulators, or probabilistic simulators which have not been smoothed to account for unseen events). Another common approach for evaluating user simulators, which has seen several variants applied in the literature, is to produce dialogs using the simulator in conjunction with a dialog manager, and evaluate properties of these dialogs relative to those where a human plays the same role.

For instance, Schatzmann et al. (2005) handcraft a simple deterministic dialog manager based on finite automata and use it in conjunction with simulators to produce synthetic dialogs. They then compute similarity measures between *real* and *synthetic* dialogs, which include dialog length, turn length, and the ratio of system and user actions per dialogue. The simulator is then deemed to be of good quality if the statistics were relatively similar. Alternatively, instead of computing similarity measures between real and synthetic dialogs, several approaches use explicit statistical tests to

sense to define the metric as the *mean per-utterance* probability rather than the overall probability of the data U.

compare the distributions over real and synthetic data. Cramér–von Mises criterion (Williams, 2008) and Kullback–Leibler Divergence (Janarthanam and Lemon, 2009) are two examples of such tests. A slight variant on this approach is have humans rank the quality of synthetic dialogs and to judge the consistency of their scores (Singh et al., 2000; Ai and Litman, 2008).

While the principle of this general approach is arguably sound, there are a number of confounding factors for such an approach. The first is that it assumes that a dialog manager *already* exists. While this may be true for many slot-filling domains (certainly those whose human-to-machine dialog data was used to train the user model in the first place), this is not the case for most domains. For example, for models induced from human-to-human dialogs, and also when addressing new kinds of domains, there is no reason to assume that a dialog manager already exists for the task. Furthermore, the user models (or simulators) are built for the purpose of training a dialog manager, and assuming that one already exists presents us with a circular argument.

A second confounding factor for this approach is that it assumes that inferences drawn about the behaviour of a simulator with a single dialog manager (or perhaps even a selection) generalise to all possible dialog managers. Finally, the form of evaluation is indirect, slow and not amenable to automatic optimisation. It sits in an uneasy place between an intrinsic metric and a task based evaluation, and arguably possesses few of the strengths and most of the weaknesses of both.

3.5.3 Evaluating learned policies

This final means of evaluating simulators can be considered the default task-based evaluation. Since most simulators are used to train dialog systems, the idea behind this evaluation is that better simulators should lead to better dialog managers when used to train them. Therefore, the evaluation strategy is to hold the dialog manager fixed in all ways *except for* the simulator, produce different versions of the dialog manager using different simulators, and then evaluate the dialog managers. Schatzmann et al. (2007b) adopt this idea, by using a simulator to train a statistical dialog manager and then evaluating the learned policy. Since the ultimate purpose of the simulator is to train a dialog manager, this is a reasonable task-based evaluation. However, there are some issues associated with this kind of evaluation that should be considered, some of which are particular to dialog management and some of which are general concerns with task based evaluations.

The primary issue is that dialog management itself is not a task for which evaluations are typically well defined or well posed, outside of exposing the system to real human users. The issue with using humans for evaluation is that it is costly, time consuming and noisy: large amounts of data need to be collected to make accurate determinations of improvement. Furthermore, which measure should be targeted is not clear, as there are several: task completion rate; dialog length; user satisfaction; etc. In any case, for system development (let alone training) using real humans is very difficult.

Without a real human, or ideally group of humans, the problem of evaluation becomes considerably harder. Since the dialog manager is by definition an interactive system, testing it in isolation is not possible. However, the only way to simulate interaction with the system is the same user simulator that was under test. Thus there is a circularity inherent to evaluation in this domain which it is difficult to break. We therefore propose that while ultimate evaluation of the dialog manager has to be the test of simulators, with the difficulty of performing this evaluation it is almost a necessity to be able to evaluate the simulators in isolation.

3.6 Conclusion

In this chapter, we reviewed the user modelling literature as a continuum: at one extreme, user models are constructed as systems driven by procedures and explicit rules to produce utterances that resemble that of users; at the other extreme user models are formal probabilistic models which treat user utterances as observations arising from a stochastic process. In the next chapter we will present the theory behind our own approach, which sits firmly at the probabilistic, data-driven extreme, while at the same time incorporates the goal-directed nature of procedural approaches.

We described the three classes of approaches commonly used for evaluating user models, namely: evaluating the predictions of the user models, evaluating synthetic dialogs resulting from the interaction of user models with a dialog manager, and finally, evaluating the dialog management policy learnt by interacting with the user model. We highlighted the strengths and weaknesses of each of the approaches of evaluation. In the next chapter, we will describe our own approach for evaluation, which advocates evaluating user models in isolation from the dialog managers they are designed to help induce in the first place, and which borrows techniques and ideas from the machine learning evaluation literature.

Chapter 4

Goal-Directed Users as a Data-Driven Generative Model

This chapter describes the approach we use for our own work in this thesis. We describe our approach broadly, illustrating its advantages over previous approaches to the problem of user modelling, specifically those presented in the previous chapter. Formal details of our methods are deferred to the appropriate chapters following this one.

4.1 Approach

We approach the problem of modelling user behaviour in task-oriented dialog domains as an instance of probabilistic modelling, with an explicit model of user goals, and which we induce entirely from data without prior access to the domain's vocabulary. Our approach can be understood to have four distinguishing properties; it is:

1. data-driven
2. probabilistic
3. generative
4. explicitly goal-directed

We argue that each of these properties offer us theoretical and practical advantages when compared to the alternative approaches in the literature discussed in Chapter 3, which were either fully probabilistic and induced from data but lacked an explicit model of user goals (Georgila et al., 2005a, 2006), or which had an explicit model

of user goals, but were not driven by real dialog data, whether probabilistic (Pietquin, 2004) or procedural in nature (Scheffler and Young, 2002; Schatzmann et al., 2007b,a; Schatzmann and Young, 2009).

4.1.1 Data-driven modelling

Adopting an empirical, data-driven approach alleviates much of the manual effort typically involved in the construction of procedural user simulators for training and testing MDP-based dialog managers. Furthermore, basing our models on empirical observations removes the need for us to speculate about how users might behave under various circumstances, which allows us to place fewer (biasing) assumptions about user behaviour than the handcrafted alternatives. Additionally, learning user models from data facilitates their use as tools for exploratory analysis of user behaviour, which would help dialog system engineers cater for the range of behaviour that human users exhibit when constructing dialog systems.

4.1.2 Generative probabilistic modelling

Probabilistic modelling allows us to express notions of variability and uncertainty in a mathematically principled way. It also allows us to separate the problems of model design, model induction, and inference: a separation which allows for a whole raft of machine learning algorithms to be deployed irrespective of the underlying model. It also makes the models more domain-independent and far more easily extensible.

The use of generative probabilistic models in particular offers us further advantages. Generative models define , not only the probability of observing a particular behaviour, but of generating the behaviour; in other words, they can be used to both predict and generate (i.e., *simulate*) user behaviour. Generative models are designed to provide complete distributions over behaviour, with a probability proportional to the behaviour's plausibility, making them suitable for capturing the range of behaviour that users can exhibit, which can vary between the highly plausible and frequent to the rare and infrequent. The distributions can also be manually inspected for a better understanding of how users behave under various circumstances.

The ability of generative models to both assign a probability to behaviour and to generate behaviour offers new avenues for evaluation, which we explore in later sections. Like other probabilistic models, they are able to assign probability to held-out test-sets, as an indicator of how probable a model finds unseen data, enabling the com-

parison of the performance of models relative to one another, but additionally, using the model in generation mode allows us to assess the quality of its output.

Furthermore, generative models define the joint probability over all model variables. This means that they can be used for discriminative tasks, such as classifying user behaviour, predicting how users might respond in a particular situations, or assigning probability to new behaviour. However, they can also be used for generative tasks, such producing the learnt behaviour, or making inferences about latent variables, such as hidden user goals, given only observed behaviour. Therefore, defining a joint distribution of all model variables offers greater flexibility (over the discriminative alternative) of how the model can be used, which is particularly important for the exploratory nature of much of the dialog research.

Finally, generative models are highly extensible, and can be adapted to accommodate the growing complexity of problems, something we demonstrate in the next two chapter. For example, variables can take different forms (strings, topics, feature vectors) depending on the nature of the input data, and additional variables can be defined in the model as needed.

4.1.3 Explicit model of user goals

To account for the goal-directed behaviour which users exhibit, we explicitly model user goals. We demonstrate that as a result of this choice of model we are better able to capture user behaviour than models which lack an explicit model of user goals.

4.2 Evaluation metrics

The models we propose in this thesis have many pragmatic benefits in terms of the kinds of behaviours they can display, as we describe above. However, in addition to this, user simulation has received little attention as a stand-alone problem in part because of how cumbersome it has proven to be to evaluate it. Thus one of the contributions of this work, and one of the strengths of our proposed approach, is in how simple it is to evaluate as a stand-alone component rather than being so tightly coupled with a dialog manager or other system.

Thus far, goal-directed user models have been partially deterministic and have required some hand-engineering. As a result, it has only been possible to evaluate them extrinsically using dialog managers. This is circular because we need simulators to

train managers, but need managers to evaluate simulators. The issue is that judgements of quality of each depend on the specifics of the other and that a proper evaluation of one depends on the correct functioning of the other. Furthermore, there is little reason to assume that because a simulator performs well with a certain dialog manager, it would perform similarly well with other managers.

In contrast, a probabilistic formulation such as we propose allows us to evaluate our models intrinsically using standard machine learning metrics, and without reference to a dialog manager, thus breaking the circularity, and guarding against such experimental biases.

To evaluate the models we adopt the use of some of the prediction-based metrics presented in Section 3.5.1.2. These metrics are based on notion that the higher the probability a model assigns to held out data (not used to train the model) the better the model is at predicting / generating the data. Because some instances will have not been encountered at training time, this measures the generalisability of the model. One such metric is the probability a model assigns to the data,where higher is better. A second (but highly related) metric is perplexity, which computes how surprising a model finds the data, and where lower is better. Both metrics have previously been used to evaluate goal-free, data-driven user models in the literature (Georgila et al., 2005a); however, here we demonstrate that the use of these metrics can be extended to evaluate a range of more complex models of user behaviour. We compute the per-utterance probability of held-out data, instead of the per-dialog probability, noting that the latter was pointed out to be incompatible across dialogs of different lengths by Pietquin and Hastie (2013). Perplexity is $2^{-log_2(d)}$ where d is the probability of the instance in question.

Additionally, we go beyond these intrinsic metrics to some extrinsic metrics which we propose, whose details we defer to the relevant chapters.

4.3 Conclusion

In this chapter, we introduced our own approach to modelling and evaluating goal-directed user behaviour in task-oriented dialog domains, which combines some of the ideas in the literature which were presented in Chapter 3. The modelling framework which we present will be used to develop models in the remainder of the thesis. In this framework, user behaviour is treated as an instance of generative probabilistic modelling, with an explicit model of user goals to capture goal directed behaviour, and

induced entirely from data without handcrafting or prior access to the domain's vocabulary. We highlighted some of the strengths of our approach compared to previous approaches, which included generality, adaptability to new domains, learnability from data, and its ability to capture not only observable dialog behaviour but also latent structure.

Finally, we motivate evaluating the models as a component isolated from dialog managers, since the user models are themselves built to help induce dialog managers. We described the evaluation metrics and approaches adopted in this thesis which are based on, and inspired by, much of the machine learning evaluation literature.

Chapter 5

Modelling Users with Latent Categorical Goals

This chapter advocates the use of a generative model to simulate user behaviour in traditional slot-filling (or informational) domains (e.g, flight and bus information domains). User goals in these domains take categorical values which allow multiple surface realisations (e.g., New York, NYC). Because the data is collected over noisy channels, surface realisations are further corrupted by speech recognition errors (e.g. New York, Newark). We describe the most commonly used representation of user-goals in the slot-filling literature: a string-based representation, and demonstrate its limitations in producing plausible user behaviour. We then present our own novel approach which adopts a topic model representation of user goals in conjunction with a bigram model to capture simple conversational dynamics. Our topic model captures both synonymy and phonetic confusability in a unified model, which we demonstrate is better able generate plausible user behaviour. We apply our modelling to two standard dialog resources: the DARPA Communicator (flight information) and Let's Go (bus information). We evaluate the models intrinsically using held-out probability and perplexity, and demonstrate substantial gains over the alternative string-based goal representation of goals, and over a (non-goal-directed) pure bigram model. To further demonstrate the strength of our model, we define the task of classifying real dialogs (consistent dialogs) and dialogs comprised of real turns sampled from different dialogs (inconsistent dialogs). We show that the classifier defined over features derived from our model lead to substantial improvement over strong baselines in performing this task. The work presented in this chapter was published in **Eshky et al. (2012)**.

5.1 Introduction

Automatically simulating user behaviour in human-machine dialogs has become vital for training statistical dialog managers in task-oriented domains, and in particular managers based on the Markov Decision Process (MDP) (Young et al., 2010). These managers are often trained with some variant of reinforcement learning (Sutton and Barto, 1998), where optimal behaviour is sought or learnt through the exploration of the space of possible dialogs. Although learning by interacting with human subjects is a possibility (Gašić et al., 2011), it has been argued that user simulation avoids the expensive, labour intensive, and error-prone experience of exposing real humans to fledgling dialog systems (Eckert et al., 1997), and instead substantial effort can be saved by training managers on simulators and only finally refining or evaluating them on real users.

Training effective dialog managers should benefit from exposure to properties exhibited by real users. Table 5.1 shows an example dialog taken from one of the domains which we consider, where the objective is to simulate the user's utterances at the semantic level. In such task oriented domains, the user has a *goal* (in this case, to book a flight from New York to Osaka), and the machine is tasked with fulfilling it. Notice that the user is consistent with this goal throughout the dialog, in that they do not provide contradictory information (although an ASR error is present), but that every mention of their destination city uses a different string. This motivates our first desideratum: that simulation be consistent with respect to the user's goals over the course of a dialog. Furthermore, one can imagine users not always responding identically in identical situations: we thus additionally require *plausible variability* in responses, while being consistent with the user goals.

Automatic tools such as speech recognisers and semantic parsers are able to process large corpora, first into text, and then into a semantic representation, which captures notions of meaning or content while abstracting away from some of the specifics of human languages. Although this kind of processing provides us with a good level of abstraction at which to model behaviour, automatic speech recognition (ASR) and semantic parsing are not error-free; speech can be incorrectly transcribed resulting in ASR noise, while semantic parsers can produce incorrect semantic representations of the text. Both sources of noise make the problem of modelling user behaviour a challenging task.

Speech	Semantic Representation
M: Hello, how can I help?	GREET META_REQUEST_INFO
U: a–a trip from New York City to Osaka, please	PROVIDE *orig_city*=**New York City** ◇ PROVIDE *dest_city*=**Salt Lake City** ⋆
M: What day would you like to travel from New York City to Salt Lake City?	IMPLICIT_CONFIRM *orig_dest_city* REQUEST *depart_date*
U: no, no– I'm leaving from New York to Osaka in Japan	ANSWER_NO PROVIDE *orig_city*=**New York** ◇ PROVIDE *dest_city*=**Osaka, Japan** ⋆
M: Sorry, leaving from New York to Osaka Japan, is that correct?	EXPLICIT_CONFIRM *orig_city* EXPLICIT_CONFIRM *dest_city*
U: Yes	ANSWER_YES

Table 5.1: An example of a slot-filling dialog in speech (left) and semantic equivalent (right) taken from the DARPA Communicator corpus. The machine (M) is tasked with eliciting the human user's (U) travel information. The data is collected at the speech level, which is automatically transcribed into text using a speech recogniser, then semantically parsed to produce *semantic utterances* shown on the right. A semantic utterance consists of an ACT, *slot*, and **Value**. We consider resources where user goals are latent, and are expressed using multiple surface realisations (◇) and corrupted by speech recognition errors (⋆). User goal annotations and gold-standard transcriptions of the speech are not available.

5.1.1 Latent user goals and corpora noise

Key to ensuring consistency of user behaviour is the notion of a *user goal* for the dialog. To further motivate our problem, let us consider the formal definition of user goals. In task-oriented domains, users are goal-directed in that they have a concrete notion of what they wish to accomplish from the dialog. For example, in a flight information provision domain, a user might wish to travel from New York City to London on the 7th of April, a construction which is referred to as a user goal, and often represented in the following form:

$$\{\textit{orig\_city}=\textbf{New York City}, \textit{dest\_city}=\textbf{London}, \textit{depart\_date}=\textbf{17-06-14}\}$$

If we assume that users express their goals by placing them in a semantic utterance of the form:

$$\text{PROVIDE } \textit{dest\_city}=\textbf{London}$$

then determining the user goals becomes a matter of extracting slot-value pairs from the semantic utterances. In reality, extracting a user goal of this form is far from a trivial task: the dialog corpora are usually collected automatically over noisy channels and without explicit feedback from the users about their true goal. Speech recognisers (and semantic parsers) make incorrect decisions about the user input, and this results in corrupted semantic user utterances. For example we might observe (as a result of a speech recognition error) the following utterance in the same dialog:

$$\text{PROVIDE } \textit{dest\_city}=\textbf{Oakland}$$

We refer to noise arising from speech recognition errors as *phonetic confusability*.

Consider the dialog in Table 5.1, taken from the DARPA Communicator corpus, a flight information provision domain. The speech is shown, as well as (a simplified version of) the final output of a semantic parser, which we refer to as a Semantic Representation. Note how the user utterance (marked with $\star$) which express the destination **Osaka** was mis-recognised by the automatic speech recogniser as **Salk Late City** (possibly due to 'Osaka, please' sounding somewhat like 'Salt Lake City'). This is an example of noise being introduced to the data due to the incorrect decision made by the automatic speech recogniser. Annotating large corpora with ground-truth output is firstly too expensive, and secondly too time-consuming to be a feasible option. When working from noisy unlabelled data, it becomes essential to be able to make inferences about the user goals with access only to their observed surface realisations.

User goals in this instance become hidden, and our approach is to treat them as latent variables in a probabilistic model[1].

5.1.2 Variability in expressing the latent user goal

In addition to the notions of latent user goals and phonetic confusability, a final point to consider relates to the surface realisations of the underlying user goals. Surface realisations exhibit a degree of variability which arises from the ability to express the same underlying goal in multiple different ways. Consider in Table 5.1 the utterances marked with ⋄: **New York** and **New York City** are two ways of expressing the same underlying flight origin, but other examples from the same corpus include **JFK**, **NYC**, or even **Manhattan** (despite not being *cities*). While these strings may be phonetically distinct, they can be considered as synonyms, or plausible variations of the same user goal, in the context of the slot concerned. We refer to the variability arising from multiple ways of expressing the same underlying goal as *synonymy*, and aim to address it in our modelling.

5.2 Overview of approach

To reiterate, we wish to automatically induce (from large noisy dialog corpora) generative models of goal-directed user behaviour, whose user goals have the following properties:

1. They are latent (true user goals are unobserved as part of the dialog).
2. They are realised as string literals which admit multiple surface realisations (e.g. Osaka, Osaka Japan).
3. They are automatically transcribed using a speech recogniser resulting in speech recognition errors (e.g. Osaka, Salt Lake City).

Our approach is to dealing with these points is as follows:

1. We treat each user goal as a latent topic indicator in a topic model, whose observations are string literal samples drawn from the topic model. The topic model allows us to cluster together co-occurring expressions of user goals, in order to

[1] We only address the noise at the level of values, and defer the treatment of noisy dialog acts / slots to future work.

account for multiple surface realisations (synonymy) and ASR noise (phonetic confusability).

2. We treat conversational dynamics between the user and the machine as arising from a simple N-gram model.

We demonstrate the efficacy of our approach on two slot-filling tasks: flight information provision (the DARPA Communicator corpus), and bus route information provision (Let's Go corpus), and compare our approach to two others. The first is a standard bigram model as first conceived by Eckert et al. (1997) and later automatically induced by Georgila et al. (2005b). The second is (our own) probabilistic formulation of goal-directed simulators, whose user goals are formulated as string literals, as envisaged by Scheffler and Young (2002), Pietquin (2004), and Schatzmann et al. (2007b). We demonstrate substantial improvement over these models in terms of predicting held-out data on two standard dialog resources: the DARPA Communicator corpus (Levin et al., 2000a; Georgila et al., 2005b) and the Let's Go corpus (Black and Eskenazi, 2009).

To further evaluate our model, we define the extrinsic task of classifying consistent dialogs from inconsistent dialogs. We show that training a classifier over features derived from our model allows us to perform accurately on this task, and to outperform a strong baseline.

The results from both sets of experiments demonstrate that latent variable models of sufficient complexity can help us achieve the best of both worlds (i.e. procedural goal-directed user modelling and stochastic data-driven user modelling): higher-order, global constraints on behaviour in a whole dialog combined with the stochastic nature and interpretability of a fully generative probability model.

5.3 Models of users in dialog

Our aim is to induce a model of user behaviour in dialogs where user goals are unobserved as part of the dialog. We consider a dialog to be a sequence of semantic utterances belonging either to the machine or the user, denoted m and u respectively. Each user utterance consists of an act a, a slot s, and a value v, while each machine utterance consists only of an act and slot. There are no restrictions on the number of contiguous machine or user utterances, i.e. machine and user utterances do not necessarily alternate in a strict order. An example of a dialog from a slot filling domain is

shown in Table 5.1.

5.3.1 N-gram models

The simplest model that can be defined to describe user behaviour in such dialogs takes into account only observable behaviour and makes no assumptions about latent states. One such model is the N-gram model of Eckert et al. (1997), which was later induced from data by Georgila et al. (2005a, 2006). The **Bigram model** (a first order Markov model) describes user behaviour as being conditioned on preceding machine behaviour. For example, the probability of a user utterance under the Bigram model would be defined as:

$$p(u_i|m_{i-1}) \tag{5.1}$$

In the simple formulation described here, each user utterance u_i, which is the complete $\{$act slot value$\}$ triple, is dependent only on the machine utterance immediately preceding it (m_{i-1}). Although Eckert et al. (1997) suggest that a wider dialog contexts can be incorporated by conditioning on a longer history beyond the preceding machine utterance, Georgila et al. (2006) find no large gains to increasing the Markov horizon (i.e. trigrams and 4-grams, etc.).

In the Bigram model, the probability of a dialog is defined as the product of the probabilities of each of the user utterances conditioned on the machine utterance preceding them, and conditionally independent from one another:

$$p(\mathbf{u}|\mathbf{m}) = \prod_i p(u_i|m_{i-1}) \tag{5.2}$$

This Bigram model (and similarly a higher order N-gram model) captures some of the conversational dynamics between the user and the machine. However, it ignores other factors contributing to the behaviour of users. We have earlier described user behaviour as being goal-directed: users behave according to a set of goals which describe what they wish to accomplish from the dialog. The Bigram model (and higher order Markov models) do not consider user goals.

5.3.2 Goal-directed models

One way to ensure goal-directed user behaviour is to explicitly model user goals. User User goals can then be incorporated probabilistically into the N-gram model by conditioning user utterances not only on preceding machine utterances but also on the user

goals in the following manner:

$$p(u_i|m_{i-1}, g) \tag{5.3}$$

Previous work has defined user goals as a set of values assigned to slots prior to the start of the dialog, and have defined values to be string literals (Scheffler and Young, 2002; Pietquin, 2004). For example, in the flight information domain, if the user wants to travel from New York City to Osaka on the 7th of April, then this is represented as:

$$g = \{orig\_city\text{=}\textbf{New York City}, dest\_city\text{=}\textbf{Osaka}, depart\_date\text{=}\textbf{07-04-14}\} \tag{5.4}$$

The probability of a dialog in terms of the observable variables can be defined by marginalising out the user goals in the following manner:

$$p(\mathbf{u}|\mathbf{m}) = \sum_g p(g_s) \prod_i p(u_i|m_{i-1}, g) \tag{5.5}$$

The simplest variant of g has string values for each of the slots the user is required to provide in order for the dialog to succeed. Thus we may have:

$$g = \{orig\_city\text{: }\textbf{New York}; dest\_city\text{: }\textbf{Osaka}\} \tag{5.6}$$

as presented in Schatzmann et al. (2005) and Schatzmann et al. (2007b). However, in these simulators, while the goal is probabilistic, there is no distribution over utterances given the goal because utterances are assembled deterministically from a series of rule applications, thus there is also no marginalisation over the goal as in (5.5) above.

The issue with a model of user goals as strings in this fashion is that users describe the same values in multiple ways (**Osaka Japan**, **Osaka**), and speech recognition errors corrupt consistent user input (**Osaka** mis-recognised as **Salt Lake City**). Users also might legitimately switch their goals mid-dialog. Inference in the model would have to allow for these possibilities: we would have to marginalise over all possible goal switches.

5.3.3 Topic-Goal model

To motivate our proposal, consider that over the course of a particular dialog one could look at the set of all the string values uttered for some slot as a count vector. For example, in some dialog, the count vector of string values uttered for the destination city slot could be:

$$\mathbf{v}_{\text{dest\_city}} = \textbf{Salt Lake City}\text{:2; }\textbf{Osaka}\text{:1; }\textbf{Osaka, Japan}\text{:1} \tag{5.7}$$

The above vector may arise because the user wants to go to **Osaka**, but the destination is initially mis-recognised as **Salt Lake City**, and the user finally disambiguates with the addition of the country. Such situations are common in the large noisy dialog resources which we consider. A naive string-based goal formulation would necessarily consider the different renderings of values to have arisen from different underlying goals.

Our approach is to treat $\mathbf{v_s}$ as a vector of string samples drawn from a topic model. More specifically, we define a Mixture-of-Multinomials topic model for each possible categorical slot s in the corpus[2]. In each model, a latent topic indicator z_s is sampled once per dialog d, and a string value v_s is sampled (conditionally on the topic indicator z_s) once per utterance i, provided that the appropriate slot s has been sampled in utterance i according to the action model (recall that we learn one such model for each slot, and the appropriate model is called upon depending on the slot sampled). Figure 5.1 shows a directed graphical representation of the Mixture of Multinomials topic model, and we define one such model for each slot independently. The topic model is defined over the *latent* user goals for a particular slot, and the associated *observed* string values.

Whilst by far the most popular topic model is the Latent Dirichlet Allocation (LDA) (Blei et al., 2003), it provides too much flexibility for our modelling task. In the original formulation of LDA, a topic indicator is sampled once per *word* instead of once per *document*. For our task, this would be analogous to sampling a topic indicator once per *utterance* instead of once per *dialog*. However, Because there are far fewer samples of slot values in a dialog than there are words in a document, LDA provides too flexible a distribution over count vectors to be used for our task. We confirmed the poor suitability of the original LDA formulation in preliminary experiments.

We now turn to a description of how the parameters to our model are estimated, and focus on how the resulting model assigns probability to dialogs. In our formulation, the latent goal for each slot is an indicator for a topic in a topic model. Each topic can in theory generate any string (so the model is inherently smoothed), but most strings in most topics will have only the smoothing weight, and most probability mass will be placed on a small number of highly correlated strings (strings that frequently co-occur with the topic). We treat the slots as being independent of one another, and thus the

[2]We class Date and Time as‘ordinal slots’ and not ‘categorical slots’.

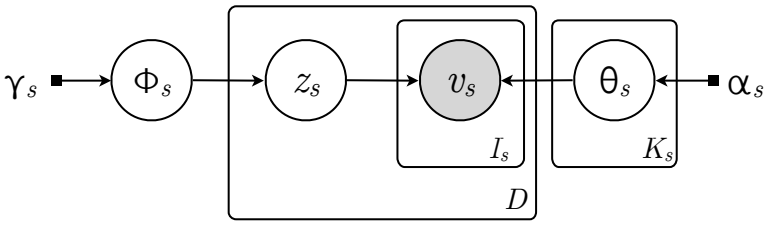


Figure 5.1: The Bayesian Mixture-of-Multinomials topic model, which defines how the observed slot values v_s are generated from the latent user goals. We learn one such model for each (categorical) slot s in the corpus. A topic indicator z is sampled once per dialog D. Each string value v is sampled once per utterance I. Word-topic parameter θ is sampled once per mixture component K, while mixture proportions parameter ϕ is sampled once globally (at the corpus level). We assume uniform Dirichlet priors over θ and ϕ, parameterised by α and γ respectively.

probability of a user goal is defined as:

$$p(g) = \prod_s p(z_s) \tag{5.8}$$

where z_s is the topic indicator for some slot s. In other words, the probability of a user goal is the product of the probabilities of the topic indicators for each possible categorical slot. If slot s has a count vector $\mathbf{v_s}$ associated with it (like the example shown in Equation 5.7) then the distribution over the values used for each slot becomes:

$$p(v_s) = \sum_{z_{s'}} p(v_s|z_s)\, p(z_s) \tag{5.9}$$

In practice, some slots will not have corresponding values, or will be slots whose values cannot be appropriately represented using the model we have defined in Figure 5.1. Dates and times, for example, have ordinal and structural relations between them, and a model which treats them as disconnected entities is not suitable. For utterances defined over categorical slots we use the topic-based goal model, and for slots with ordinal, structural, or no values, we use the Bigram model defined over the entire semantic utterance as in Equation (5.2). A better treatment of such slots is deferred to future work.

To model the interaction between the utterances of the user and the machine, we define an **Act Model**, which describes the probabilities of the dialog acts and slots in a user utterance. The Act Model is bigram-based, but is defined only over $\{$ACT, *slot*$\}$ pairs, compared to the original Bigram Model, which was defined over the entire

user utterance consisting of the {act, slot, value} triple. Under the Act Model, the probability of a user utterance under the Act Model would be defined as:

$$p(a_i, s_i | m_{i-1}) \tag{5.10}$$

while the probability of a dialog under the Act model is defined as the product of the probabilities of each of the user utterances conditioned on the machine utterance preceding them, and conditionally independent from one another, as:

$$p(\mathbf{u}|\mathbf{m}) = \prod_i p(a_i, s_i | m_{i-1}) \tag{5.11}$$

Finally, our complete model of user utterances, the Topic-Goal Model, is defined as the probability of {ACT, *slot*}, given by the Act Model in Equation 5.10, times the probability of the values for the slots, given by Equation 5.9. The following equation defines our complete model:

$$p(\mathbf{u}|\mathbf{m}) = \prod_i p(a_i, s_i | m_{i-1}) \cdot \prod_s p(v_s) \tag{5.12}$$

5.4 Parameter estimation

5.4.1 Topic-Goal model parameter estimation

Our topic model is a Bayesian version of the Mixture-of-Multinomials model. Under this model, each dialog has associated with it a latent variable z_s for each slot s in the goal, which indicates which topic is used to draw the values for that slot. Conditioned on z, independent samples are drawn from the distribution over words to which that value of z corresponds—however, the effect in the marginal distribution over words is to strongly prefer sets which have co-occurred in training as these are assigned to the same topic.

Bayesian inference in mixture models has been described in detail in Neal (1991) and Griffiths and Steyvers (2004), so we give only a brief account here for our particular model. We take r appropriately-spaced samples from a Gibbs sampler over the posterior mixture parameters θ, ϕ: θ are the word-topic parameters and ϕ are the mixture proportions. We assume a uniform Dirichlet prior over θ and ϕ, leading to Dirichlet posteriors which we integrate out in the predictive distribution over v using the standard Dirichlet integral. For each of our r samples we have K mixture components, parameterised by γ_{rz} (the Dirichlet parameter for the z-th mixture component in

the r-th sample) and α_{rzj} for each word j in the z-th topic for the r-th sample. The $\bullet$ notation indicates a sum over the corresponding index, i.e. $\gamma_{r\bullet} = \sum_z \gamma_{rz}$. Then:

$$p(v) = \frac{1}{|r|} \sum_r \sum_z \frac{\gamma_{rz}}{\gamma_{r\bullet}} p(v|\alpha_{rz}) \tag{5.13}$$

$$p(v|\alpha) = \frac{\Gamma(\alpha_\bullet)}{\Gamma(\alpha_\bullet + v_\bullet)} \prod_j \frac{\Gamma(v_j + \alpha_j)}{\Gamma(\alpha_j)} \tag{5.14}$$

This states that each of the r samples has topics z which are multinomial distributions with posteriors governed by parameters α_{rz}. For any of these topics, the distribution over v is as given in Equation (5.14) (we suppress the subscripting of α here for the different samples and topics, since this holds whatever its value). The final predictive probability given in Equation (5.13) averages over the samples r and the topics z (with topics weighted by their parameters γ_{rz}).

5.4.2 Bigram model parameter estimation

Because we require a distribution over all possible utterances, assigning non–zero probability to cases outside of the training data, our Bigram model is interpolated with a unigram model, which itself is interpolated with a smoothing model which assumes independence between the act, slot, and value elements of the utterance. Interpolation weights are set to maximise probability of a development set of dialogs. Each sub-model uses the maximum likelihood estimator (the relative frequency of the utterance), and unseen machine utterances place full weight on the unigram/smoothed model (ignoring the bigram probability since it has no meaning if m_{i-1} is unobserved). We label this model the Bigram model in subsequent experiments.

5.5 The dialog resources: DARPA Communicator and Let's Go

We conduct our experiments on two standard slot-filling, human-to-machine dialog resources, where our objective is to model the human's behaviour, conditioned on the machine's (or system's) behaviour at the semantic level.

The first of the two resources is the DARPA Communicator corpus (DC), which is a collection of dialogs concerning a flight information provision domain. The DARPA Communicator was collected between 2000-2001 through the interaction of human users with 10 different dialog systems (Levin et al., 2000a). The semantic form of

Corpus	Semantic Utterance
DC	PROVIDE_INFO *orig_city* **Boston**
LG	INFORM *place* [departure_place **CMU**, arrival_place **airport**]

Table 5.2: Example of semantic utterances from the two corpora: the DARPA Communicator (DC) and Let's Go (LG). Note how in addition to the value, the Let's Go utterances contain properties (e.g., departure_place and arrival_place).

the corpus was later automatically produced by Georgila et al. (2005b) by parsing the natural language form of the dialogs.

The second dialog resource is Let's Go (LG), which is a collection of dialogs concerning a bus information provision domain. Let's Go was collected through the interaction of the CMU dialog system, which provides bus arrival and departure information, with real users, who were trying to find their way through the city of Pittsburgh. We use years 2007, 2008, and 2009, which were distributed as part of the Spoken Dialog Challenge (Black and Eskenazi, 2009).

For both dialog resources we model at the automatically produced semantic level, which contains noise arising from automatic speech recognisers (and possibly from the semantic parsers, although, as previously mentioned, we currently only explicitly handle speech recognition noise). We also model the user goals, which are not annotated as part of either corpora, by treating them as latent variables in our models.

5.5.1 Corpora preprocessing

We preprocessed the corpora by converting Communicator XML-tagged files and Let's Go system log files into sequences of ACT, *slot*, and **value** utterances. Table 5.2 shows some examples of the utterances after preprocessing. We divided the corpora into training, development, and test sets as follows: Communicator contains 2285 dialogs in total, and Let's Go contains 17992, and in each case we selected 80% of dialogs at random for training, 10% for development, and 10% for testing.

We note that Let's Go is a far noisier resource than the Communicator, as it contains more variability in slot values. The greater variability might be due to the fact that users can be more flexible with their bus routes than their flight destinations. Another possibility is that Let's Go dialogs were collected over noisier channels, which results in a larger percentage of speech recognition errors. In addition, we note that Let's Go semantic parses contain a kind of ambiguity not present in Communicator: the parser

fails to distinguish departure from arrival places over 90% of the time, and instead assigns them a generic Single_Place property. Our models currently only handle uncertainty about the value and assume that the decisions made by the semantic parser are correct, but an improvement could incorporate semantic noise into the model. We defer this more complex treatment to future work.

5.5.2 Setting the model parameters

Free model parameters are set by a simple search on the development set, where the objective is likelihood—for the Bigram Model the parameters are the interpolation weights, and for the topic model we search for the number of topics and smoothing constant for the topic distributions. For Let's Go, since we can have multiple places provided in a single act, we treat each utterance as containing a set of values and build the count vector for the topic model as the union of these sets over the whole dialog. The slots over which the topic model is defined for Communicator are *dest_city* and *orig_city* (this takes into account PROVIDE and REPROVIDE acts). For Let's Go we derive the model over the three properties: single_place, arrival_place and departure_place, as opposed to the less informative slot *place*.

5.6 Intrinsic evaluation of models

Our first evaluation metric is an intrinsic, information theoretic metric, based on the notion that better models find new data (unseen at training time) to be more predictable. One such metric is the probability a model assigns to test data. A second metric is perplexity, which tells us how surprised a model is by the data. We argue that these metrics are more suitable than the precision and recall metrics previously used, because they acknowledge that rather than each user response being "correct" at the point at which it is observed, there is a distribution over possible responses.

The held-out probability metric should be understood as a means of comparing the relative viability of different models of the same data. Note that we are reporting the probability of unobserved data, rather than data from which the models were induced, and are thus measuring the generalisability of the models. The absolute numbers are hard to interpret, as there is no hard upper bound; while it may be appealing to think of an upper bound of 1, this is incorrect as it would imply that there was no variability in the data. However, it should be understood that assigning particular behaviour higher

probability means that the model is more likely to produce it when run in simulation mode—and since the user behaviour in question has not been seen at training time, this measures the extent to which the models have generalised beyond the training data relative to one another. This metric is closely related to perplexity, commonly used to evaluate language models, such that each can be computed from the other: the major difference is that the log-probability is on a logarithmic scale. We report the mean per-utterance log probability of unseen data, that is, the probability of the whole held-out corpus divided by the number of user utterances.

5.6.1 An upper-bound on String-Goal models

For the sake of comparison, we compute an upper-bound on String-Goal models, which gives a flavour for how such models would perform *optimistically*. The upper-bound assigns probability to dialogs as follows: for each utterance u_i if the corresponding value v_i has been seen before in the dialog, the probability used for that utterance is computed according to the Act Model (which was defined in Section 5.3.3) as $p(a_i, s_i|m_{i-1})$, that is, the probability of the act a_i and slot s_i only; there is no penalty for repetition of the value, similar to Equation 5.10. If the value is unseen in the dialog, we use the full probability of the utterance from the Bigram Model, as described in Equation 5.2. This is optimistic because there is no penalty for repeated goal changes besides that imposed by the Bigram Model itself, and no penalty or choosing between previously sampled goals as would be necessary in a probability model.

Any string-based model necessarily assigns lower probabilities to data than the upper bound, because it would penalise goal changes (in a probabilistic sense; that is, there would be a term to reflect the probability of some new goal given the old) to allow for the discrepancy in values present in dialogs. In contrast, our upper bound does not include such a term. Furthermore, once multiple goal values had been uttered in the dialog, we would have to sample one to use for the next utterance, which would again incur some cost: again, we do not have such a cost in our upper bound.

We could in theory use an external model of noise to account for these value discrepancies (and the ASR errors we model in the next section). However, this would further decrease the probability, as some probability mass currently assigned to the held-out data would have to be reserved for the possibility of string renderings other than those we observe.

It bears mentioning that our upper bound on string-goals is not a generative model.

However, it allows us to assign probabilities to unseen data (albeit optimistically), and thus provides us with a point of comparison. Although not technically a model, we refer to this as the String-Goal model for the remainder of the chapter.

5.6.2 Results

Figure 5.2 shows the results of our evaluation. We see that the Bigram Model is weak on both resources. The results of the String-Goal model suggest that there is much variability due to synonymy and recognition errors which string-goals are unable to capture. The Topic-Goal model explains this much more easily by grouping commonly co–occurring values into the same topic.

Table 5.4 shows the perplexities corresponding to the performances with 100% training data for all acts and just PROVIDE acts (perplexity is $2^{-(lp)}$ where lp is the log probability). Improvements are more apparent when we compute the probability over PROVIDE acts alone, which the models are designed to handle. And since perplexity is not on a log scale, the differences are more pronounced.

The Act Model (defined in Equation 5.11) which is a Bigram Model over $\{$ACT, *slot*$\}$ pairs alone excluding the values, demonstrates the vast discrepancy in uncertainty between the full problem and the valueless prediction problem. We note that the perplexity of our Act Model on Communicator is comparable to that of Georgila et al. (2006). Table 5.3 shows the log probability.

5.7 Extrinsic evaluation of model

Having shown in the previous section that our Topic-Goal model is a stronger predictor of held-out data than the String-Goal and Bigram alternatives, we now turn to a task-based evaluation for a demonstration of the model's ability to capture consistency with respect to the latent user goals.

5.7.1 Evaluating model consistency

In the face of slot-value synonymy and phonetic confusability, we define *inconsistent* dialogs to be ones that are locally coherent but lack the structure of a real dialog from one turn to the next. We then suggest that an appropriate task-based evaluation for models intended to capture consistency is the task of distinguishing between consistent and inconsistent dialogs. One can imagine, for instance, that the Bigram model which

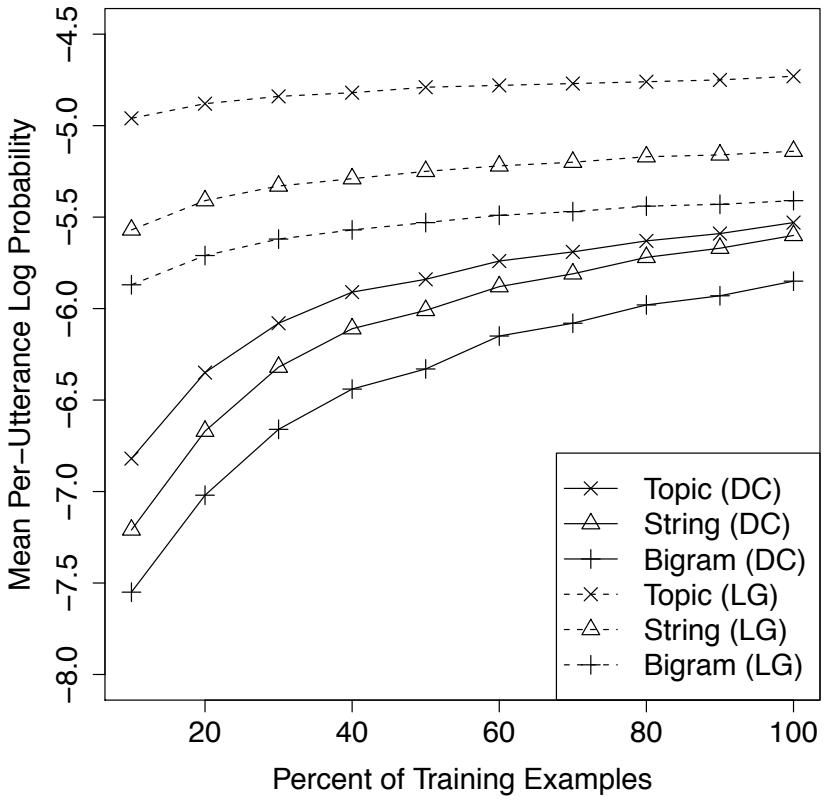


Figure 5.2: The mean per-utterance log probability, assigned by the different models to the held-out data of the two resources: DARPA Communicator (DC) and Let's Go (LG). The test data remains fixed while the percentage of training dialogs is increased. The better performance on Let's Go (LG) compared to the DARPA Communicator (DC) is explained by the sizes of the corpora, where LG is 8x larger than DC, and thus more data is available. Our model, the Topic model, outperforms the others.

Model	DC(A)	DC(P)	LG(A)	LG(P)
Bigram Model	−5.85	−8.7	−5.41	−9.22
String-Goal Model	−5.6	−7.16	−5.14	−8.43
Topic-Goal Model	−5.53	−6.76	−4.73	−7.26
Act Model	−2.26	−1.65	−1.02	−0.85

Table 5.3: The mean per-utterance log probability, assigned by the different model to the held-out data of the two resources: DARPA Communicator (DC) and Let's Go (LG). (A) is averaged over all acts, while (P) is averaged only over the PROVIDE acts, the acts which the models are designed to improve predictions of. We also report the perplexity computed by the Act Model, in demonstration of the vast discrepancy in uncertainty between the full problem and the valueless prediction. The Topic-Goal Model find the held-out data to be the most predictable.

Model	DC(A)	DC(P)	LG(A)	LG(P)
Bigram Model	347.88	5979.53	223.23	10125.87
String-Goal Model	270.09	1286.03	169.87	4578.23
Topic-Goal Model	252.78	860.2	113.45	1417.06
Act Model	9.56	5.2	2.77	2.34

Table 5.4: The mean per-utterance perplexity, computed by the different models over the held-out data of the two resources: DARPA Communicator (DC) and Let's Go (LG). (A) is computed over all acts while (P) is computed only on PROVIDE acts: the acts which the models are designed to improve predictions of. We also report the perplexity computed by the Act Model, in demonstration of the vast discrepency in uncertainty between the full problem and the valueless prediction. The Topic-Goal Model shows the highest reduction in perplexity.

was designed to capture only local coherence, to be of no help in this task, but that goal-directed models should be of help. To test this hypothesis, we define the following classification problem: using features derived from the Topic-Goal model which we have automatically induced in the previous sections, can we build a classifier that would discriminate between real dialogs (consistent dialogs) and dialogs comprised of real turns sampled from different dialogs (inconsistent dialogs)? In this section, we demonstrate that we can indeed perform this task with high precision when building a classifier over features derived from our model. Similar task-based evaluation has been proposed in the discourse coherence literature (Elsner and Charniak, 2008).

To answer this question, we first need two sets of dialogs: a set of consistent dialogs and a set of inconsistent dialogs. We then need to determine how useful our model is in addressing this question compared to alternative models. While we can compare the utility of our model to that of the String-Goal model, the Bigram model by definition provides no help in this task, and is thus excluded from this comparison.

5.7.2 Classification datasets

We create a classification dataset as follows: original dialogs, taken directly from the corpus with no alternations, are the positive examples in our classification problem. As for negative examples, we require an equal number of dialogs to positive examples, and also require the dialogs to be locally coherent but to lack the structure of a real dialog. To create these negative or "fake" examples we do the following: first we sample whole turns ($\{$machine, user$\}$ pairs) at random from the designated corpus. We then keep a histogram over real dialog lengths, and sample a number of turns for our "fake" dialogs proportional to this histogram. We then sample this many turns from the frequency distribution over turns in the real data, and create exactly as many dialogs in this fashion as real dialogs (as positive examples) in the data. The result is an equal number of dialogs to real dialogs comprised of real turns, of (expected) real length, but where the sequence of turns is highly unlikely to be coherent given the random sampling. Thus, the classification problem is far from trivial.

We conduct this experiment on the Communicator corpus. We produce positive and negative examples from the training dataset to produce training examples for the classifiers, and from the development dataset to provide test instances.

Feature Set	Features
String Consistency (S)	Did the user provide a different value for the dest_city?
Dialog (S)	Dialog length (turns)
	Mean, standard deviation, min and max acts per turn
	Presence of special machine acts (flight offer and confirm)
	Presence of user acts (provide a dest city and arrival city)
	Proportion of acts which were PROVIDE acts
	Did the user provide a different value for the orig_city?
Topic-Goal (T)	Ranked list of posterior probabilities of top 50 topics
	Normalised probability of dialog for topic model

Table 5.5: For the task-based evaluation of classifying consistent from inconsistent dialogs, we train classifiers over three feature sets. The "String Consistency" features are binary features indicating whether different values have been provided for the same slot. Based on our consistency hypothesis, the String Consistency feature set is anticipated to be the weakest. The "Dialog" feature set contains dialog-specific features and is deliberately chosen to be a strong baseline. The "Topic Model" features are simply taken from our model. The results of the classification task are shown in Table 5.6 and Figure 5.3.

5.7.3 Classification features

We train linear Support Vector Machine (SVM) classifiers using the features described in Table 5.5. These feature sets are designed to capture different aspects of consistency. The first set of features we derive is intended to replicate the utility of string-based goals: we set up binary features which indicate whether different information is provided for the slots over the course of the dialog. We refer to these features as the String Consistency features. In this instance, different surface realisations of the same value would be deemed inconsistent.

We derive a second set of features, the Dialog features, are intended to capture surface level features of the dialogs, inspired by the "similarity measures" presented in the work of Schatzmann et al. (2005), which provided features to compare *simulated* dialogs to real dialogs. Our setting is different: we do not seek to tell real dialogs from simulated ones, but real dialogs from permuted versions of real dialogs. In addition to length-based features, we add binary presence indicator for several user and machine acts highly correlated with the completion of dialogs, as well as acts which indicate

the provision of information and the proportion of all acts which were provision acts. Table 5.5 gives a complete list of these Dialog features. Recall that the Bigram model is excluded from this experiment, and that Dialog features do not correspond to the Bigram Model.

Finally, we use our model to derive the Topic-Goal consistency features. These features are the posterior distribution over topics for each slot given a dialog. Our topics were induced from the real training dialogs, and their posterior probabilities are computed for all dialogs relative to this model. We take posterior probabilities of the fifty most probable topics for each of the *dest_city* and *orig_city* slots as features, as well as the normalised log probability of the dialog (the log probability divided by the number of user utterances). These form our Topic-Goal features.

We train linear SVM classifiers for this task, and we use LIBSVM (Chih-Chung Chang and Chih-Jen Lin, 2011), scaling features to the range $[0-1]$. All other parameters are left at their defaults. Table 5.5 gives the complete list of features.

5.7.4 Classification results

The results of the classifiers are shown in Table 5.6, and the same results are depicted graphically in Figure 5.3. Since we have a balanced binary classification task, accuracy is the most appropriate metric. Here we see that the classifier trained on the Topic-Goal model features alone outperforms the classifiers trained on the Dialog features and the String Consistency features. Combining the Dialog features and String Consistency features improves the overall performance over the individual feature sets but does not outperform Topic-Goal features alone. The inclusion of all three feature sets buys us significant gains in performance when compared to the String Consistency and Dialog features combined, indicating that the Topic-Goal model learns a useful notion of consistency of values from noisy unlabelled dialogs which is not shared with the information from the two other feature sets.

The results presented in this section demonstrate that the model we have induced encodes notions of consistency which go beyond those defined at the level of strings. Features derived from the latent Topic-Goal space have substantially improved performance in a non-trivial classification task, demonstrating that our model captures an important notion of how real dialogs appear, which is not shared by the other feature sets which we have considered.

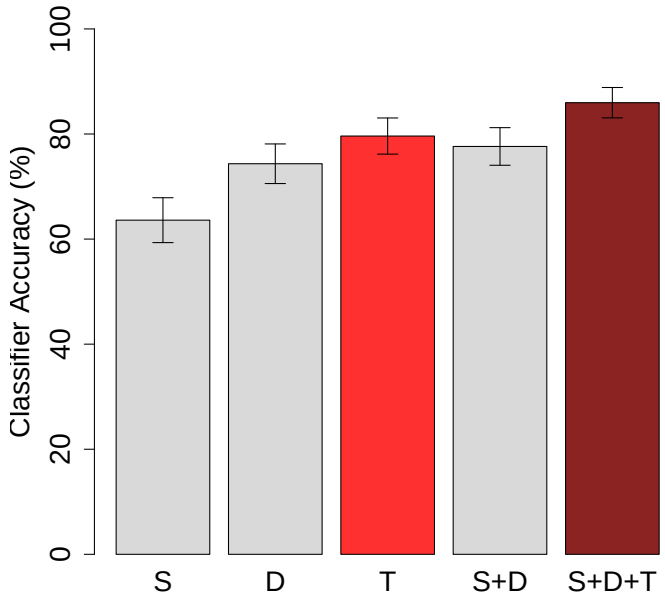


Figure 5.3: The performance of five linear SVM classifies trained on the feature sets shown in Table 5.5. D=Dialog features, S=string consistency features, and T=Topic-Goal model features. Errors are 95% intervals to the accuracies assuming they are parameters to a binomial distribution. Results demonstrate that the classifier trained on T features alone outperforms the classifiers trained on the strong D features and the weak S features individually. Combining D and S improves the overall performance over the individual feature sets but does not outperform T alone. The addition of T features to D + S buys us further gains in performance, indicating that the Topic-Goal model, which we have induced in this chapter, learns a useful notion of consistency of values from noisy unlabelled dialogs.

Classifier Feature Set	Classifier Accuracy
String Consistency (S)	63.60 ± 4.27
Dialog (D)	74.34 ± 3.77
Topic-Goal (T)	79.61 ± 3.44
S + D	77.63 ± 3.58
S + D + T	85.96 ± 2.89

Table 5.6: The performance of five linear SVM classifies trained on the feature sets shown in Table 5.5. Errors are 95% intervals to the accuracies assuming they are parameters to a binomial distribution. Results demonstrate that the classifier trained on T features alone outperforms the classifiers trained on the strong D features and the weaker S features individually. Combining D and S improves the overall performance over the individual feature sets but does not outperform T alone. The addition of T features to D + S buys us further gains in performance, indicating that the Topic-Goal model, which we have induced in this chapter, learns a useful notion of consistency of values from noisy unlabelled dialogs.

d	z_{dest_city} [probability]	proportion of samples	user utterance given topic z_{dest_city} and machine utterance REQUEST_INFO *dest_city*
1	**Norfolk Virginia** [0.562]	0.264	PROVIDE_INFO *dest_city* **Norfolk Virginia**
	Norfolk [0.234]	0.111	PROVIDE_INFO *dest_city* **Norfolk**
	Newark Virginia [0.088]	0.039	PROVIDE_INFO *dest_city* **Newark Virginia**
	Virginia Beach [0.0412]	0.028	PROVIDE_INFO *orig_city* **Las Vegas Nevada**
	Newark [0.040]	0.028	NO_ANSWER *null* **no**
		0.025	COMMAND *start_over* **start over**
2	**Chicago** [0.350]	0.164	PROVIDE_INFO *dest_city* **Chicago**
	Chicago Illinois [0.182]	0.082	PROVIDE_INFO *dest_city* **Chicago Illinois**
	Duluth Minnesota [0.124]	0.057	PROVIDE_INFO *dest_city* **New Orleans**
	New Orleans [0.122]	0.055	PROVIDE_INFO *dest_city* **Duluth Minnesota**
	New Orleans Louisiana [0.085]	0.039	PROVIDE_INFO *dest_city* **New Orleans Louisiana**
		0.028	NO_ANSWER *null* **no**
3	**Anchorage** [0.539]	0.252	PROVIDE_INFO *dest_city* **Anchorage**
	Anchorage Alaska [0.148]	0.072	PROVIDE_INFO *dest_city* **Anchorage Alaska**
	Jacksonville Florida [0.124]	0.056	PROVIDE_INFO *dest_city* **Jacksonville Florida**
	Great Anchorage Alaska [0.098]	0.048	PROVIDE_INFO *dest_city* **Great Anchorage Alaska**
	Duluth Minnesota [0.057]	0.047	PROVIDE_INFO *orig_city* **Hartford Connecticut**
		0.026	PROVIDE_INFO *dest_city* **Duluth Minnesota**

Table 5.7: Samples drawn from the Topic-Goal model. Left: top 5 strings with associated probabilities sampled from topics for three different dialogs *d*. Right: top 6 utterances (plus fraction of samples in 10,000) generated in response to the machine utterance "REQUEST_INFO *dest_city*" and conditioned on the topic z_{dest_city}.

5.8 Generating model behaviour through sampling

In this section we give examples of the output of our Topic-Goal model in generation mode (i.e. simulation mode), which corresponds to sampling from the model. Our examples are drawn from the model induced for the Communicator data. Sampling from standard distributions is available in most statistical environments, or can be implemented following the algorithms in Bishop (2006) and other statistical resources. We sample utterances by drawing ACT-*slot* pairs from the distribution:

$$p(a_i, s_i | m_{i-1})$$

which corresponds to drawing a value from a multinomial distribution. If we sample a PROVIDE_INFO act, then we check whether we have sampled a topic for the corresponding slot thus far in the dialog. If not, then we sample one by drawing a topic indicator from $p(z_s) = \frac{\gamma_{rz}}{\gamma_{\bullet\bullet}}$ and then drawing a multinomial distribution over strings from the Dirichlet posterior corresponding to z. Once the topic for the slot is set, we sample **values** as draws from the fixed multinomial and add these to the ACT-*slot* pair.

Table 5.7 shows some examples drawn from the model. For each row in the table (corresponding to a new dialog d), we sample a topic for the *dest_city* and *orig_city* as needed, and sample 10000 utterances given that topic. The left hand side of the table shows the top five strings in the sampled topic, while the right hand side shows the top six utterances in response to REQUEST_INFO *dest_city*. Note that the proportion of utterances on the right does not match the probability of the values on the left because of the presence of other user acts besides PROVIDE *dest_city*.

5.9 Conclusion

In this Chapter, we presented a complete generative probabilistic modelling framework of goal-directed user behaviour in informational, or slot-filling, dialog domains. We showed how to induce models within this framework for two different dialog domains: a flight information domain, and a bus route information domain, applied to two standard dialog resources (DARPA Communicator and Let's Go). The probabilistic goal-directed models which we presented are the first of their kind in the literature to be fully induced from data.

We focussed on a sub-problem which arises in most slot-filling dialog corpora: user goals in slot-filling domains are categorical entities which allow multiple surface

realisations (e.g. Osaka, Osaka Japan), and because the dialogs are collected over noisy channels, surface realisations are further corrupted by speech recognition errors (e.g. Osaka, Salt Lake City). Furthermore, the dialogs are presented in the corrupted form without gold-standard human annotations, and lack user goal annotations; thus, we were faced with the challenging task of inducing goal-directed user models with only observable behaviour. To address this problem, we treated the string values observed in the dialogs as samples drawn from a latent topic model, namely a Mixture-of-Multinomials topic model. We induced a different topic model for each categorical slot (e.g. arrival_place, departure_place, flight_origin, etc.). We then incorporated the novel goal model into a larger generative model which captures the conversational dynamics as simple Bigram Model (which conditions the choice of $\{$Act Slot$\}$ pair of the user on the previous $\{$Act Slot$\}$ of the machine). We called our complete model the Topic-Goal Model.

We applied our modelling to two task-oriented dialog domains, a flight information domain, the DARPA Communicator corpus, and a bus route information domain, the Let's Go corpus, and evaluated it intrinsically by demonstrating that the Topic-Goal Model is better able to predict held-out data than a simple Bigram Model with no notion of consistency, and an upper-bound on probability models where the goals are represented as string. We measured this using held-out probability (as a log-probability) and Perplexity. We then move on to show in a task-based evaluation that features derived from the model lead to substantial improvement in the task of discriminating between real dialogs (consistent dialogs) from dialogs where the turns have been selected at random from the training data (inconsistent): this is a fairly difficult task, but our model allows significant improvement over strong baselines.

The model presented in this chapter could be extended in a number of ways. It could be improved to incorporate the noise resulting from the decisions made by the semantic parser. Another possible improvement is to explore the effects of introducing dependency between the slots in the user goal, which would enforce more plausible values pairings and would potentially improve the simulator's performance. The effects of a dependence assumption between the different utterances occurring in a single user turn under the Act Model can also be explored. And finally, a more sophisticated conversational dynamics model can replace the simple bigram-based interaction component in our model.

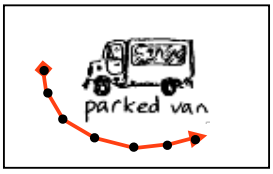
Chapter 6

Modelling Users with Spatial Goals

This chapter advocates the use of a generative probabilistic model to simulate user behaviour in a task-oriented dialog domain, not previously attempted for automated dialog management or user simulation, namely: the Map Task, where user goals are spatial routes across artificial landscapes. We show how to derive an efficient feature-based representation of spatial goals which admits exact inference and generalises to new routes. The use of a generative probabilistic model allows us to capture a range of plausible behaviour given the same underlying goal. We induce the model automatically from data, and evaluate it intrinsically using held-out probability and perplexity, and find a substantial reduction in uncertainty brought by our spatial representation. We then evaluate extrinsically first by computing accuracy scored against an exact match to human responses (accuracy), and then in a human judgement task, in which Mechanical Turk workers score the appropriateness of a route description (produced by our model or by real human Givers). We find, in this human judgement task, find that our model's behaviour does not differ significantly from the behaviour of real users. Some of the work presented in this chapter has been published in **Eshky et al. (2014)**.

6.1 Introduction

Automated dialog management is an area of research that has undergone rapid advancement in the last decade. The driving force of this innovation has been the rise of the statistical paradigm for monitoring dialog state, reasoning about the effects of possible dialog moves, and planning future actions (Young et al., 2013). Statistical dialog management treats conversations as Markov Decision Processes, where dialog moves are associated with a utility, estimated online by interacting with a simulated



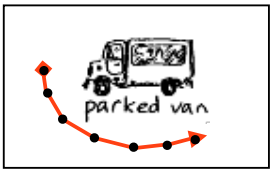
Instruct MOVE(UNDER, LM)

Figure 6.1: A sub-route can be seen as a user goal, which as raw at a is presented numerically as a series of pixel coordinates, but is conveyed using a semantic utterance, a disparity not present in traditional slot0-filling domains.

user (Levin et al., 1998; Roy et al., 2000; Singh et al., 2002; Williams and Young, 2007; Henderson and Lemon, 2008; Young et al., 2010). However, most of this research has been confined to dialog in slot-filling domains, with exceptions which include work on troubleshooting domains (Williams, 2007) and relational domains (Lison, 2010, 2013)).

Although navigational dialogs have received much attention in studies of human conversational behaviour (Anderson et al., 1991; Thompson et al., 1993; Reitter and Moore, 2007a,b), they have not yet become the subject of automated dialog management research, and existing systems addressing spatial navigation, such as Spacebook (Janarthanam et al., 2013), remain largely handcrafted. Other existing systems, such as MATCH (Johnston et al., 2002), NJFun (Singh et al., 2002), and Parlance (Gasic et al., 2013), which operate within what could be classed as navigation problems, exclude any spatial reasoning and treat the problem as simple slot-filling. Navigational domains present an interesting challenge, due to the disparity between the spatial goals and their grounding as utterances. This disparity renders much of the statistical management literature inapplicable, because previous approaches have been designed to deal with goal representations that are identical to their grounding as utterances (in particular, within slot-filling domains). In this chapter, we address this deficiency.

We focus on the task of simulating user behaviour, both because of the important role simulators plays in the induction of dialog managers, and because it provides a self-contained means of developing the domain representations which facilitate dialog reasoning. We show how a generative model of user behaviour can be induced from data, alleviating the manual effort typically involved in the development of simulators, and providing an elegant mechanism for reproducing the natural variability observed in human behaviour.



$u_1 = \text{MOVE}(\text{UNDER}, \text{LM})$
$u_2 = \text{MOVE}(\text{TOWARDS}, \text{ABSOLUTE\_EAST})$

Figure 6.2: Each spatial route (illustrated with a red line) can be described using multiple different navigational units (e.g. u_1 or u_2). As raw data, a route is presented numerically as a series of pixel coordinates (marked in black dots).

6.1.1 Spatial goals of users

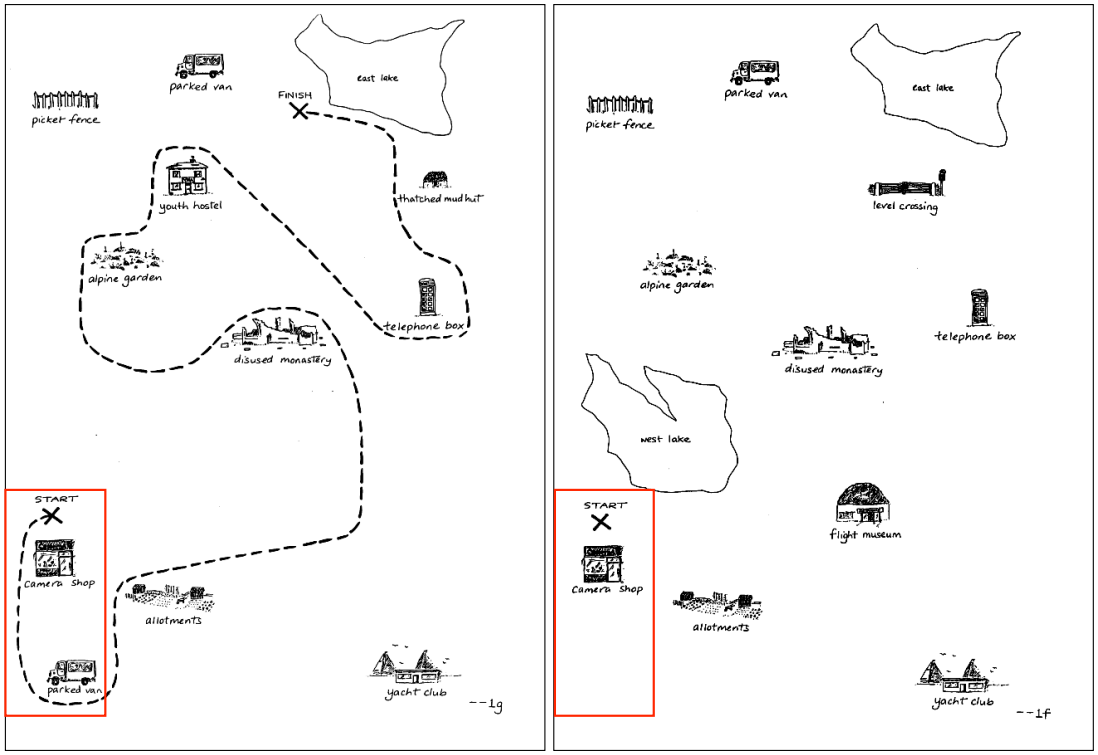
As we have previously mentioned, users in task-oriented domains are goal-directed, with a persistent notion of what they wish to accomplish from the dialog. In slot-filling domains, user goals are comprised of a group of categorical entities, represented as slot-value pairs. These entities can be placed directly (or *slotted*) into the user's utterance. For example, in a flight booking domain, if a user wishes to fly to London from New York on the 17$^{\text{th}}$ of June, then their goal can take the following form:

$$\{\textit{orig\_city}=\textbf{New York City}, \textit{dest\_city}=\textbf{London}, \textit{depart\_date}=\textbf{17-06-14}\}$$

while expressing the destination can take the following form:

$$\text{PROVIDE } \textit{dest\_city}=\textbf{London}$$

In contrast, consider the task of navigating somebody across a landscape. Figure 6.3 shows a pair of maps taken from a spatial navigation domain, the HCRC Map Task corpus, where the task is for the instruction Giver to communicate a route to a Follower, whose map may differ. Because the Giver aims to communicate their route, which is not visible to the Follower, one can view the route as the Giver's goal for the dialog. However, unlike goals in slot-filling domains, it is unclear whether the route can be represented categorically in a form that would allow the Giver to communicate it by placing it directly into an utterance. As raw data, a specific route is presented numerically as a series of pixel coordinates, and at the dialog level is expressed using a semantic utterance. Figure 6.1 shows an example. Before modelling interlocutors in this domain, we must derive a meaningful representation for the spatial goals, and then devise a mechanism of mapping these spatial routes to the semantic utterances which express them.



A Giver's map Corresponding Follower's map

Figure 6.3: An example of a pair of maps from the HCRC Map Task corpus. In the Map Task, the instruction Giver's task is to communicate a route to a Follower, whose map may differ. The route can be seen as the Giver's goal which the Follower tries to infer. A corresponding dialog (concerning only part of the route inside the red box shown in this map) is given in Table 6.1.

Natural Language	**Semantic Representation**
G: you are above the camera shop	*Instruct* POSITION(ABOVE, LM)
F: yeah	*Acknowledge*
G: go left jus– just to the side of the paper, • then south, under the parked van ○ you have a parked van?	*Instruct* MOVE(TO, PAGE_LEFT) • *Instruct* MOVE(TOWARDS, ABSOLUTE_SOUTH) *Instruct* MOVE(UNDER, LM) ○ *Query-yn*
F: a parked van no	*Reply-n*
G: you go– you just go west, • and down, and then you go along to the– you go east ○	*Clarify* MOVE(TOWARDS, ABSOLUTE_WEST) • *Clarify* MOVE(TOWARDS, ABSOLUTE_SOUTH) *Clarify* MOVE(TOWARDS, ABSOLUTE_EAST) ○
F: south then east	*Check*
G: yeah	*Reply-y*

Table 6.1: A Giver (*G*) and a Follower (*F*) alternating turns in a dialog concerning the maps in Figure 6.3. The utterances are shown in natural language (left), and the semantic equivalent (right), which is composed of *Dialog Acts* and NAVIGATIONAL UNITS. Utterances marked • demonstrate a plausible variability in expressing the same part of the route on the Giver's map, and similarly those marked ○. We model the Giver's behaviour, conditioned on the Follower's, at the semantic level. (Route descriptor TOWARDS indicates a movement in the direction of the referent ABS_WEST, whereas TO indicates a movement until the referent is reached).

6.1.2 Utterance variability for the same goal

In addition to being goal-directed Givers tend to express the same underlying route in multiple different ways. Consider the dialog in Table 6.1, resulting from the maps in Figure 6.3. Utterances marked ∘ (and similarly those marked •) illustrate how the same spatial route can be described in different ways. Specifically, the Giver first expresses the movement from the start point to the left by referencing a page element: MOVE(TO, PAGE_LEFT), and a second time by referencing an absolute direction: MOVE(TOWARDS, ABS_WEST)[1]. The Giver also expresses the movement under the parked van by referencing position relative to a landmark: MOVE(UNDER, LM),and a second time as a movement referencing an absolute direction: MOVE(TOWARDS, ABS_EAST).

It is crucial to note that the variability in the descriptions of the route is not only at the natural language level, but also at the semantic level. For example, the two navigational units, MOVE(UNDER, LM) and MOVE(TOWARDS, ABSOLUTE_EAST) are semantically unrelated, but in the context of the dialog describe the same part of the underlying route, as Figure 6.2 illustrates. A model providing a 1-to-1 mapping from spatial routes to semantic utterances would fail to capture this property. Instead, we need to be able to account for plausible variability in expressing the underlying spatial route as semantic utterances.

A different kind of variability would be the linguistic variability in expressing the same navigational unit. For example, the navigational unit MOVE(TOWARDS, ABSOLUTE_SOUTH), can be linguistically realised as “move south” or as “go down”. This kind of linguistic variability is typically a property of the Natural Language Generation Component (NLG), a problem beyond the scope of this thesis[2].

6.2 Overview of approach

In order to perform efficient reasoning, we propose a new feature-based representation of spatial goals, transforming them from coordinate space to a low-dimensional feature space. This groups similar routes together intelligently, permitting exact inference, and generalising to new routes. To address the problem of variability of utterances given

[1] Route descriptor TOWARDS indicates a movement in the direction of the referent ABS_WEST, whereas TO indicates a movement until the referent is reached.

[2] Note that in the previous chapter (Chapter 5), the surface realisation of user goals were conveyed as arguments for the slots in the semantic utterances.

the same underlying route, we learn a distribution over possible utterances given the feature vector derived from a route, with probability proportional to the plausibility of the utterance.

Because this domain has not been previously addressed in the context of dialog management or user simulation, there is no directly comparable prior work. We thus conduct several novel evaluations to validate our model. We first use intrinsic information theoretic measures, which compute the extent of the reduction in uncertainty brought by our feature-based representation of the spatial goals. We then evaluate extrinsically by generating utterances from our model, and comparing them to held-out utterance of real humans in the test data. We also utilise human judgements for the task, where the judges score the output of the different models and the human utterances based on their suitability to a particular route.

6.3 The model

The task is to model the Giver's utterances in response to the Follower's, at the semantic level. Using the original dialog act annotations (Anderson et al., 1991), embedded with a semantic representation (Levit and Roy, 2007), a Giver's utterance, which instructs the Giver to move in a particular direction, takes the form:

$$g = Instruct,\ u = \textsc{Move}(\textsc{Under}, \textsc{LM})$$

consisting of a dialog act g and a navigational unit u[3]. Aligned with the navigational unit u, is an ordered set of waypoints W, corresponding only to part of the route u describes. Figure 6.4 shows an example of such a sub-route drawn on a map. The point-set W can be seen as the Giver's current goal on which they base their behaviour. Because the routes are drawn on the Giver's maps, we treat W as observed.

To model some of the interaction between the Giver and the Follower, we additionally consider in our model the previous dialog act of the Follower, which could for example be:

$$f = Acknowledge$$

Given point-set W and preceding Follower act f, as the Giver, we need to determine a procedure for choosing which dialog act g and navigational unit u to produce. In other

[3]We align g and u in a preprocessing step. We also store the names of the landmarks which the navigational units abstract away from.

words, we are interested in the following distribution:

$$p(g,u|f,W) \tag{6.1}$$

which says that, as the Giver, we select our utterances on the basis of what the Follower says, and on the set of waypoints we next wish to describe.

To formalise this idea into a generative model, we assume that the Giver act g depends only on the Follower act f. We further assume that the navigational unit u depends on the set of waypoints W which it describes, and on the Giver's choice of dialog act g: This provides a simple way of incorporating the different sources of information into a complete generative model. Using applications of Bayes' Rule and some independence assumptions, we arrive at the following generative model[4]:

$$p(g,u|f,W) = \frac{p(u)\ p(g|u)\ p(f|g)\ p(W|u)}{\sum_{g'u'}\ p(u')\ p(g'|u')\ p(f|g')\ p(W|u')} \tag{6.2}$$

This equation requires four distributions: $p(u)$, $p(g|u)$, $p(f|g)$, and $p(W|u)$. The first three constitute the semantic component of our model, to which we dedicate Section 6.3.1, while the fourth constitutes the spatio-semantic component, to which we dedicate Sections 6.3.2–6.3.4.

6.3.1 The semantic component

The semantic component concerns only the categorical variables, f, g, and u, and addresses how the Giver selects their semantic utterances based on what the Follower says. We model the distributions u, $g|u$, and $f|g$ from Equation (6.2) as categorical distributions with uniform Dirichlet priors:

$$u \sim Cat(\alpha) \qquad \alpha \sim Dir(\varepsilon) \tag{6.3a}$$

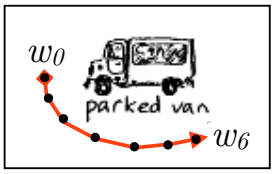
$$g|u \sim Cat(\beta) \qquad \beta \sim Dir(\kappa) \tag{6.3b}$$

$$f|g \sim Cat(\gamma) \qquad \gamma \sim Dir(\lambda) \tag{6.3c}$$

6.3.2 Spatial goal abstraction

Each ordered point-set W on some given map can be seen as the Giver's current goal, on which they base their behaviour. Let $W = \{w_i; 0 \leq i < n\}$, where $w_i = (x_i, y_i)$ is a waypoint, and x_i, y_i are pixel coordinates on the map, typically obtained through a vision processing step.

[4]The derivation of the model is in Appendix A



$$u = \textsc{Move}(\textsc{Under}, \text{LM})$$

Figure 6.4: In the data, each of the Giver's navigational units u, is aligned with an ordered set of waypoints W, corresponding to a sub-route.

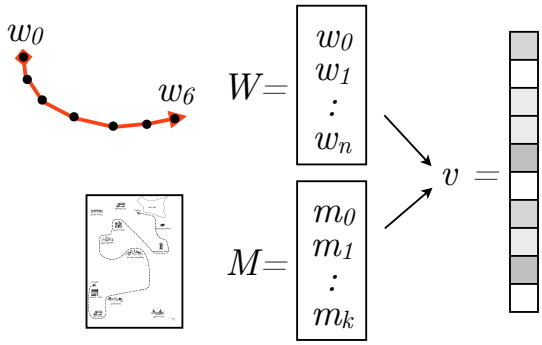


Figure 6.5: We extract a spatial feature vector v of fixed length, from point-sets W and M, which are of varying lengths. W is the ordered set of points corresponding to a spatial sub-route, while M is the unordered set or points indicating landmark locations and map boundary information.

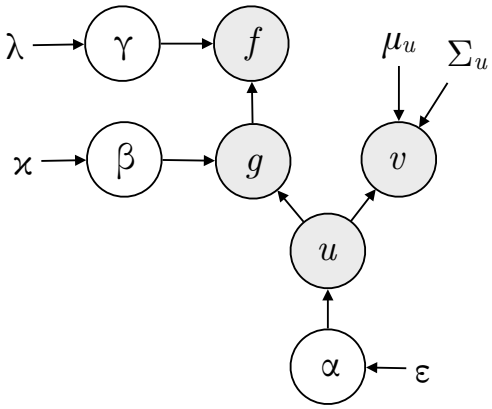


Figure 6.6: We define a generative model of the Giver's semantic utterances, over the Giver's dialog act g and navigational unit u, preceding Follower act f, and spatial feature vector v. Latent parameters and priors are shown. Note that the arrows only indicate conditional probabilities rather than direct causal links.

Given this goal formulation, from Equation (6.2) we require $p(W|u)$, i.e. the probability of a set of waypoints given a navigational unit. However, there are two problems with deriving a generative model directly over W. Firstly, the length of W varies from one point-set to the next, making it hard to compare probabilities with different numbers of observations. Secondly, deriving a model directly over x, y coordinates introduces sparsity problems, as we are highly unlikely to encounter the same set of coordinates multiple times. We thus require an abstraction away from the space of pixel coordinates.

Our approach is to extract feature vectors of fixed length from the point-sets, and then derive a generative model over the feature vectors instead of the point-sets. Feature extraction allows point-sets with similar characteristics, rather than exact pixel values, to give rise to similar distributions over navigational units, thus enabling the model to reason given previously unseen point-sets. The features we extract are detailed in Section 6.3.4.

6.3.3 The multivariate normal distribution

Let M be a point-set describing map elements, which include landmark locations (centre of the landmark) and map boundary information (the edge of the map), and where each position in the point-set is labelled appropriately. $M = \{m_j; 0 \leq j < k\}$, where $m_j = (x_j, y_j)$ is a map element with pixel coordinates x_j and y_j. We define a spatial feature function $\psi : W, M \rightarrow \mathbb{R}^n$ which captures, as feature values, the characteristics of the point-set W in relation to elements in M. Let the spatial feature vector, extracted from the point-set W and the map elements M, be:

$$v = \psi(W, M) \tag{6.4}$$

Figure 6.5 illustrates the feature extraction process, while Section 6.3.4 describes the features we extract. We now define a distribution over the feature vector v given the navigational unit u. We model $v|u$ as a multivariate normal distribution (recall that v is in $\mathbb{R}^n$):

$$v|u \sim N(\mu_u, \Sigma_u) \tag{6.5}$$

where μ and Σ are the mean vectors and covariance matrices respectively. Subscript $_u$ indicates that there is one such parameter for each navigational unit u.

6.3.4 The spatial feature sets

Here we define the spatial feature sets which we extract from W and M to arrive at feature vector v in Equation 6.4. We compute four different feature sets from the ordered set of waypoints W representing the sub-route, and from the point-set M containing elements of the map (as we described in Section 6.3.3). The four feature sets are:

1. **Absolute features** capture directions and distances of movement. We compute the distance between the first and last points in W, and compute the angle between unit vector <0,-1> and the line connecting first and last points in W. This gives rise to 2 features

2. **Polynomial features** capture shapes of movements as straight lines or curves. We compute the mean residual of a degree one polynomial fit to the points in W (linear), and a degree two polynomial (quadratic), which gives rise to 2 features. These features are computed quickly and efficiently, requiring only the solution to a least squares problem

3. **Landmark features** capture how close the route takes the Follower to the nearest landmark. We compute the distance between the end-point in W and the nearest landmark in M, where the nearest landmark is automatically inferred based on the coordinates. We additionally compute the angle between the route taken in W and the line connecting the start point in W to the nearest landmark. This gives rise to 2 features

4. **Edge features** capture the relationship between the movement and the map edges. We compute the distance from the start-point in W to the nearest edge and corner in M, and similarly for the end-point in W, giving rise to 4 features

The four feature sets give rise to 10 features in total, and thus the feature vector v is of fixed length 10.

6.3.5 The complete generative model

Our complete generative model of the Giver is a distribution over Giver act g and navigational unit u, given the preceding Follower act f and the spatial feature vector v. Vector v is the result of applying the feature extraction function ψ over W and M, where W is the ordered point-set describing the sub-route aligned with u, and M is

the point-set describing landmark locations and map edge information. We rewrite Equation (6.2) as:

$$p(g,u|f,v) = \frac{p(u)\ p(g|u)\ p(f|g)\ p(v|u)}{\sum_{g'u'}\ p(u')\ p(g'|u')\ p(f|g')\ p(v|u')} \tag{6.6}$$

We call our model the Spatio-Semantic Model, **SSM**, and depict it in Figure 6.6.

6.4 Parameter Estimation

In this section, we describe how we estimate our model from data. For the **semantic component**, described in Equations 6.3, we use point estimates for α, β and γ, fixing them at their posterior means in the following manner:

$$\hat{\beta}_{gu} = p(g|u) = \frac{Count(g,u)+1}{\sum_{g'} Count(g',u)+L} \tag{6.7}$$

Where $L =$ size of vector β. We do the same for $\hat{\alpha}$ and $\hat{\gamma}$. As for the **spatial component**, it is a matter of estimating a multivariate normal distribution for each navigational u we encounter, as shown in Equation 6.5. Since the alignments between navigational units u and point-sets W are fully observed, parameter estimation is a question of estimating the mean vectors $\mu_{u'}$ and the covariance matrices $\Sigma_{u'}$ from the point-sets co-occurring with navigational unit u'. While a fully Bayesian estimation with tied parameters would likely help estimate accurate distributions for rarely seen utterances, for these experiments we use maximum likelihood estimators. To avoid issues with degenerate covariance matrices resulting from small amounts of data, we use diagonal covariance matrices. Because $v|u$ is normally distributed, inference, both for parameters and conditional distributions over navigational units, can be performed exactly, and so the model is exceptionally quick to learn and perform inference.

Because we also want to be able to assign probabilities to previously unseen instances, we smooth our models by learning what we refer to as a **background model**. We estimate the background model from all the training data, which results in high variance in the distribution over features and a flat overall distribution. Where no model can be estimated for a particular navigational unit, we use that navigational unit's smoothed prior probability combined with the background model for its likelihood.

6.5 The dialog resource: Map Task

We conduct our experiments on the Map Task corpus (Anderson et al., 1991) which is a collection of cooperative human-human dialogs arising from the spatial navigation task explained in Figure 6.3 and Table 6.1. The two human participants, an instruction Giver and an instruction Follower, are each given a map depicting a hand-drawn landscape. The maps are almost but not quite identical, and differ in that

- some landmarks occur only in one of the maps and not the other
- and that the Giver's map contains a route marked with start and end points

The navigational task is for the Giver to convey the route for the Follower, and for the Follower to draw (on their own map) the route which they've inferred. The corpus contains 128 dialogs concerning 15 different pairs of maps. Figure 6.3 shows and example of a pair of maps taken from the corpus, and Table 6.1 shows an example of part of a dialog arising from the pair of maps. Although we don't directly address the disparity between the Giver's map and the Follower's map in our modelling, it still results in richer dialogs, which implicitly capture the idea of different world knowledge.

In this chapter, we model the Giver's behaviour conditioned on the Follower's, and consider the spatial routes drawn on the Giver's map as their observed user goals. Despite being observed, the goals presents an interesting challenge because they are represented numerically as a sequence of pixel coordinates while being grounded as semantic utterances; a disparity not present in conventional slot-filling domains, where the goals are represented and grounded using categorical entities of the same form.

To our knowledge, there are no attempts to model instruction Givers as users in the Map Task domain. Two studies model the Follower in the context of understanding natural language instructions and interpreting them by drawing a route (Levit and Roy, 2007; Vogel and Jurafsky, 2010). Both studies exclude dialog from their modelling. Although their work is not directly comparable to ours, they provide a suitable corpus for our task.

The original release of the corpus contained dialog acts (such as *Acknowledge* and *Instruct*) in addition to natural language transcriptions of the dialogs, all of which were manually labelled by expert annotators (Anderson et al., 1991). Because we model behaviour at the semantic level, we require semantic annotations of the human-transcribed dialogs, which can typically be obtained through a semantic parse of the

natural language utterances. We also model the goals as spatial routes, whose annotations are typically obtained through vision processing of the maps.

In our experiments, we use an existing extension of the corpus by Levit and Roy (2007), which is semantically and spatially annotated by expert annotators. The spatial annotations are x, y pixel coordinates of landmark locations and evenly spaces points on the routes. All 15 giver maps from the original corpus were annotated. The navigational units take the predicates MOVE, TURN, POSITION, or ORIENTATION, and two arguments: a route descriptor and a referent. The semantic annotations were restricted to the Giver's *Instruct*, *Clarify*, and *Explain* acts. Out of the original 128 dialogs, 25 were semantically annotated.

6.5.1 Corpus statistics and state space

For our experiments, we use all 15 maps, and all 25 semantically annotated dialogs. A dialog on average contain 57.5 instances, where an instance is an occurrence of f, g, u, and W. We find 87 unique navigational units u in our data, however, according to the semantic representation, there can be 456 distinct possible values for u[5]. As for the rest of the variables, f takes 15 values, g takes 4, and v is a real-valued vector of length 10, extracted from the real valued sets W and M of varying lengths. We thus reason in a semantic state space of $87 \times 15 \times 4 = 5220$, and an infinite spatial state space.

6.6 Intrinsic evaluation of model

Our first evaluation is intrinsic, based on notion that the higher the probability a model assigns to held out data (new instances of data not used to train the model), the better the model is at predicting / generating that data. One intrinsic metric which we compute is the probability a model assigns to held-out data, or *held-out probability*. Note that while higher numbers are better, the target is *not 1* as this is not a percentage. A second metric which we compute is the related information theoretic metric of *perplexity*, which measures how surprising a model finds the held-out data, where lower is better. Perplexity is $2^{-log_2(d)}$ where d is the probability of the instance in question. Both metrics have been used to evaluate user simulators in the literature (Georgila et al., 2005a; Eshky et al., 2012).

[5] 20×2 for TURN and ORIENTATION, $+$ 208×2 for MOVE and POSITION.

Feature Set	Perplexity
Absolute (A)	7.26 ± 4.08
Polynomial (P)	12.86 ± 8.39
Landmark (L)	15.16 ± 6.27
Edge (E)	17.92 ± 8.47
All	4.66 ± 2.22

Table 6.2: Perplexity scores (and standard deviations) of our model, computed over the four feature sets and their combination, estimated through leave-one-out validation. (A) outperforms all individual sets, while the combination of all features yields the best overall performance.

The advantage of these metrics is that they allow us to evaluate the entire distribution provided by our model. They also allow us to compare the performance of models relative to one another. In our next set of experiments, we compute the per-utterance probability of held-out data, instead of the per-dialog probability (the latter was deemed incompatible across dialogs of different lengths by Pietquin and Hastie (2013)).

6.6.1 Experimental setup

We evaluate using leave-one-out validation, which estimates the model from all but one dialog, then evaluates the probability of that dialog. We repeat this process until all dialogs have been evaluated as the unseen dialog.

6.6.2 Evaluating the feature sets

We first consider the suitability of the different feature sets for predicting utterances. Figure 6.7 shows the mean per-utterance probability our model assigns to held-out data when using different sets. The more predictable the model finds the data, the higher the probability. Note that the target metric here is *not* 1, as there is no single correct answer. It can be seen that the most successful features in order of predictiveness are: Absolute, then Polynomial, then Landmark, and finally Edge. The combination of all buys us further improvement. Perplexity is shown in Table 6.2.

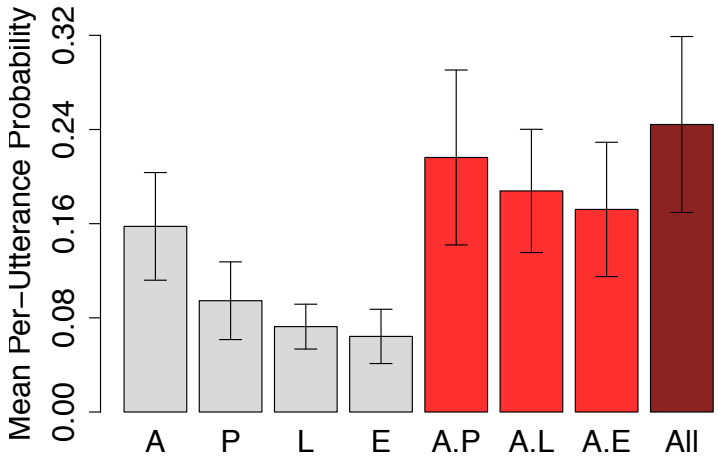


Figure 6.7: Mean per-utterance probability, assigned to held-out data by our model when defined over the four feature sets and their combinations, estimated through leave-one-out validation. A=Absolute, P=Polynomial, L=Landmark, and E=Edge. Error bars are standard deviations. The model defined over the combination of all features yields the best performance.

6.6.3 Evaluating the model

Secondly, we consider two baselines inspired by models in the literature (Eckert et al., 1997; Levin et al., 2000b; Georgila et al., 2005a). Both are variants of our model that lack the spatial component, (have no model of goal). Although the baselines are weak, they allow us to measure the extent of reduction in uncertainty brought by the addition of the spatial component to our model, which is the primary purpose of this comparison. **Baseline 1** is $p(g,u)$ while **Baseline 2** is $p(g,u|f)$. The first tells us how predictable Giver utterances are (in the held-out data), based only on the normalised frequencies. The second tells us how predictable they become when we condition on the previous Follower act. Details of the baselines are similar to equations described in Section 6.3.1.

Figure 6.8 shows the mean per-utterance probability our model assigns to held-out data when compared to the two baselines. Baseline 2 slightly improves our predictions over Baseline 1, although not reliably so, when considering the small increase in perplexity in Table 6.3. SSM demonstrates a much larger relative improvement across both metrics. The results demonstrate that our spatial component enables substantial reduction in uncertainty, brought by the transfer of information from the maps to the utterances.

Intrinsic metrics, such as the probability of held-out data and perplexity, provide

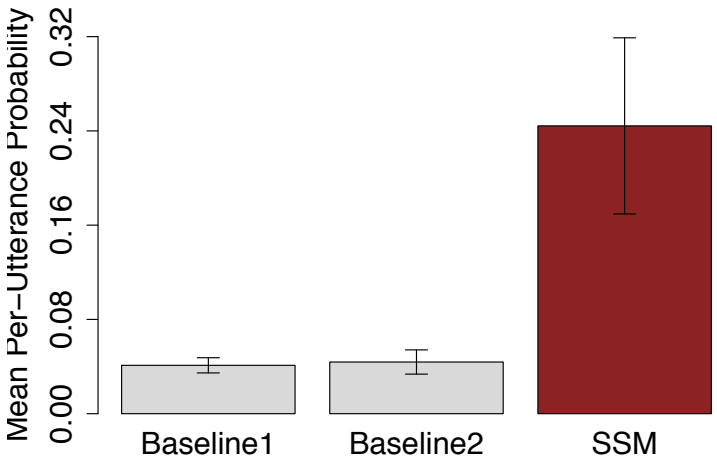


Figure 6.8: Mean per-utterance probability assigned to held-out data by our model (SSM), compared to two baselines which lack the spatial component, estimated through leave-one-out validation. Error bars are standard deviations. The large increase in performance by our model demonstrates the extent of the reduction in uncertainty brought by our spatial representation.

Model	**Perplexity**
Baseline1	24.95 ± 4.05
Baseline2	25.06 ± 12.02
SSM	4.66 ± 2.22

Table 6.3: Perplexity scores of our model (SSM) compared to the two baselines, estimated through leave-one-out validation. Our model finds the held-out data to be least surprising, and by a large margin, demonstrating the extent of the reduction in uncertainty brought by our spatial representation.

us with an elegant way of evaluating probabilistic models in a setting where there is no single correct answer, but a range of plausible answers, by exploiting the model's inherit ability to assign probability to behaviour. However, the metrics can be hard to interpret in an absolute sense, providing much better information about the relative strengths of different models rather than their absolute utility. In the next section, we explore methods for determining the utility of the models when applied to tasks.

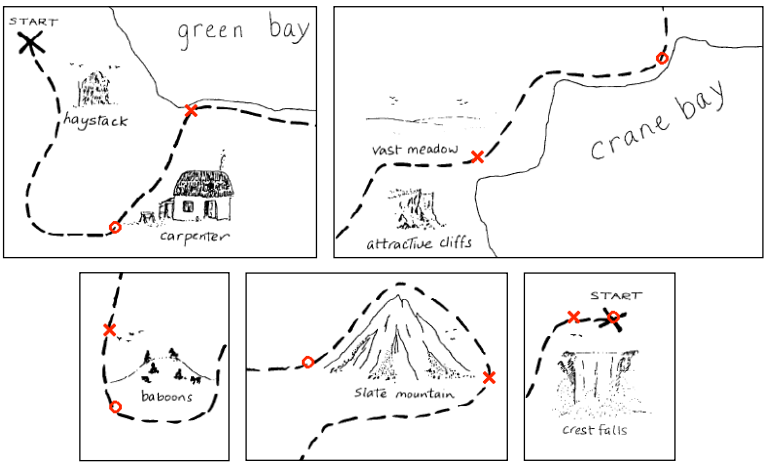


Figure 6.9: Examples of sub-routes from the held-out data for the extrinsic evaluation. Each sub-route is marked with a start and end point (○ and × in red). Each model to be evaluated produced a navigational unit to describe the sub-route.

6.7 Extrinsic evaluation of spatial component

We have shown in the previous section that our spatial component improves our predictions of the held-out data. But in the absence of an upper bound, the numerical values of the intrinsic metrics (of perplexity and held-out probability) can be difficult to interpret. To get a better intuition of the intrinsic gains, we conduct a task-based evaluation of our model output, and focus on evaluating the spatial component to investigate its usefulness.

6.7.1 Experimental setup

We train on 22 of the dialogs, holding out 3 dialogs, concerning 3 unseen maps, for testing. The reason for doing this is to implicitly demonstrate the model's ability to reason about new maps not seen at training time, as well as new dialogs. The evaluation task is as follows: for each sub-route in the held-out data, we generate from the model the most probable description of that route[6]. Figure 6.9 shows some examples of sub-routes taken from the test dialogs, and Figure 6.10 shows examples of sub-routes along with the most probable navigational unit to describe each under our model, **SSM**.

[6]The models can generate 1 of the 87 navigational units observed in the training set, but are made to output the most probable in this experiment.

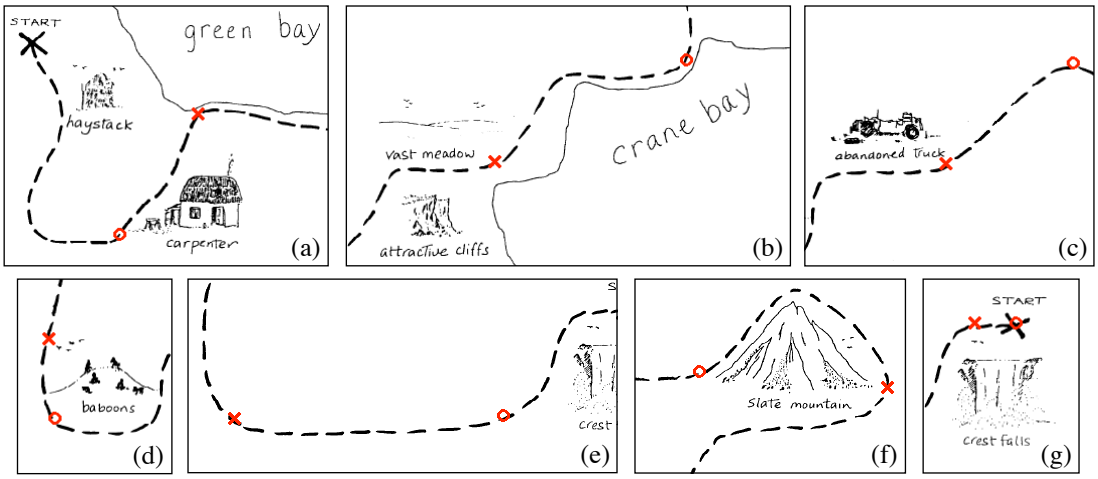


Figure 6.10: Examples from the extrinsic test set. Given a sub-route marked with start-point ∘ and end-point × (in red), our model generates the following u values: (a) MOVE(TOWARDS, ABS_NORTHEAST) (b) MOVE(FOLLOW-BOUNDARY, LM) (c) MOVE(TOWARDS, ABS_SOUTHWEST) (d) MOVE(PAST, LM) (e) MOVE(TOWARDS, ABS_WEST) (f) MOVE(AROUND, LM) (g) TURN(ABS_WEST).

6.7.2 Accuracy

We first explore a naive notion of accuracy. We treat the original Giver utterances which were used in the holdout data itself to describe the routes; these were produced by the original human Givers at the time the corpus was created, and we refer to them as **Real Giver**. We then compute the percentage of model-generated navigational unit which match the Real Giver navigational units. And here we are interested in the quality of most probable navigational unit. As a **Baseline**, we produce the most probable navigational unit under our model *unconditioned on the route*. This results in a frequency-based navigational unit, and since we are producing the most probable then this is always MOVE(TOWARDS, LM).

Although accuracy and the related metrics of precision and recall are widely used to evaluate simulators in the literature, we argue that it is an unsuitable metric as it assumes a single correct answer when many are feasible. One can expect such metrics to heavily penalise the valid variability of models. We conduct this experiment (and the one that follows as a confirmation of this idea).

A first glance at the results in Table 6.4 might suggest that both models have little utility: SSM is "correct" only 33% of the time, while the Baseline only 7% of the time. However, the extent to which this conclusion follows depends on the suitability of accuracy as a means of evaluating dialog. As we have motivated earlier in the

Model	Accuracy
Baseline	7.69%
SSM	33.08%

Table 6.4: The percentage of model-generated navigational units that match Real Giver navigational units in the test set. The models output the most probable navigational unit to describe a given sub-route. We argue that accuracy, and the related metrics of precision and recall, are unsuitable as they assume one correct answer when many are feasible.

Model	Average Score
Mismatch	1.45
Baseline	3.04
SSM	5.27
Real Giver	5.11

Table 6.5: The average scores assigned by human judges to model-generated navigational units on a Likert scale of 1-7. We treat the numerical values as continuous intervals and not as ordinal values. The deliberate Mismatch is judged to be the worst, which confirms the reliability of the data collected on mechanical turk. The Baseline, which is only based on frequencies and lacks the spatial component, has some utility and performs better than the mismatch. SSM and Real Giver are both scored reasonably well, and are judged to be of similar quality.

introduction of the chapter, it is rarely the case in dialog that there is a single correct answer and a host of incorrect ones, but rather a gradient of descriptions from the highly informative and appropriate to the nonsensical and confusing. Such subtleties are not captured by an accuracy test (or the closely related recall and precision). In demonstration of this point, we next conduct qualitative evaluation of model output.

6.7.3 Human judgement evaluation

We follow a similar experimental setup to the accuracy experiment: for each sub-route in the held-out data, we generate from *a model* the most probable description of that route. However, instead of comparing that against the Real Giver navigational units from the original corpus, we ask humans (Mechanical Turk workers) to rate, on a

Likert scale of 1-7, the degree to which a given navigational unit provides a suitable description of a given sub-route. Sub-routes are taken from the test dialogs, and are marked similarly to the accuracy experiment. Examples of the sub-routes as displayed to human judges are shown in Figures 6.11 to 6.14 (note how the entire map is shown so as to give the human judges the entire spatial context). Navigational units are generated from SSM, Baseline, Real Giver, and a control condition: a deliberate **Mismatch** to the sub-route. The Mismatch is generated automatically by taking the least probable navigational unit under SSM, of the form MOVE(TOWARD, x) where x is one of the four compass directions. The Mismatch condition is designed to assess the reliability of the judgements collected.

We collect 5 judgements for each sub-route-unit combinations on Mechanical Turk, and randomise so that no judge sees the same order of pairs. Test dialogs contained 94 distinct sub-routes, and each model outputs a single answer, excluding the Real Giver: in the original corpus, Real Givers describe the same sub-route multiple times (we account for the unbalanced experiment design when we analyse the judgements collected). This gives rise to approximately 2,000 judgments in total.

We analyse the results with a two-way ANOVA (ANalysis Of VAriance), with factors "model" and"sub-route" for a 4×94 design. The *means* of the "model" factor are shown in Table 6.5. It can be seen that Mismatch is scored very poorly, as we expect, and that the frequency-based Baseline, although unconditioned on the spatial routes still has some utility and performs better than the Mismatch. We also find that SSM and Real Giver are scored reasonably well, and are judged to be of a similar quality. We thus proceed with a more rigorous analysis.

The ANOVA summary is shown in Table 6.6. A significant effect of the model factor is present, meaning that the scores assigned by human judges to the navigational units are significantly influenced by which model was used to generate the navigational units. Additionally, a significant effect for the sub-route factor can be seen, which is due to some sub-routes being harder to describe than others. An interaction effect is also present, which is expected given such a large number of examples. Note how the model factor accounts for the largest amount of variance of all the factors.

Having confirmed the presence of a model effect, we conduct a post-hoc analysis of the model factor. Table 6.7 shows a Tukey HSD (Honest Significant Difference) test, demonstrating that all models are significantly different from one another, except Real Giver and SSM. Results show that, despite the large number of judgements collected, we are unable to separate the quality of our model's navigational unit from that in the

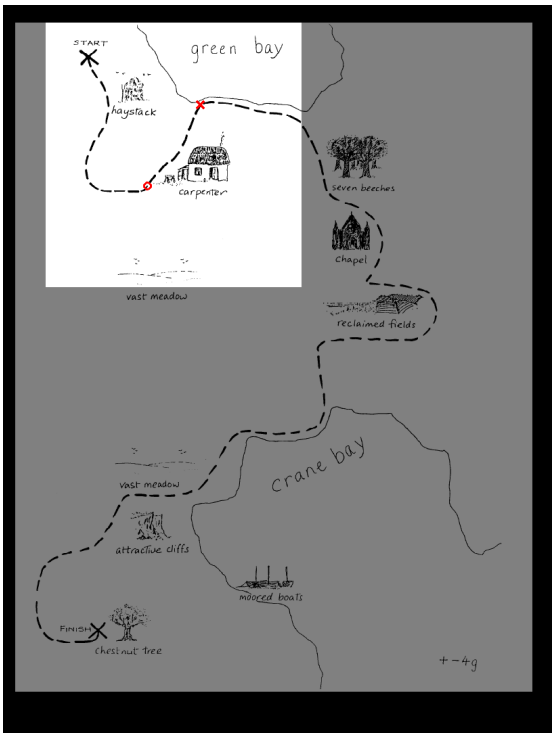


Figure 6.11: Example of a sub-route as displayed to a human judge. Each sub-route is marked with start and end points (○ and × in red).

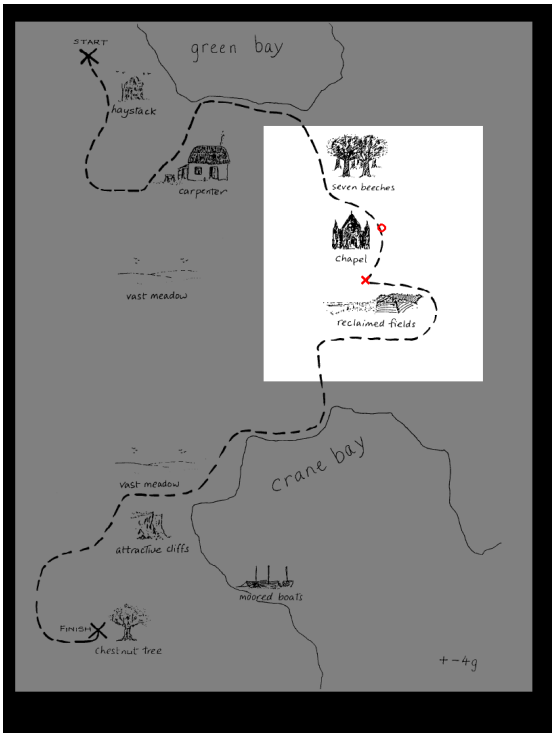


Figure 6.12: Example of a sub-route as displayed to a human judge. Each sub-route is marked with start and end points (○ and × in red).

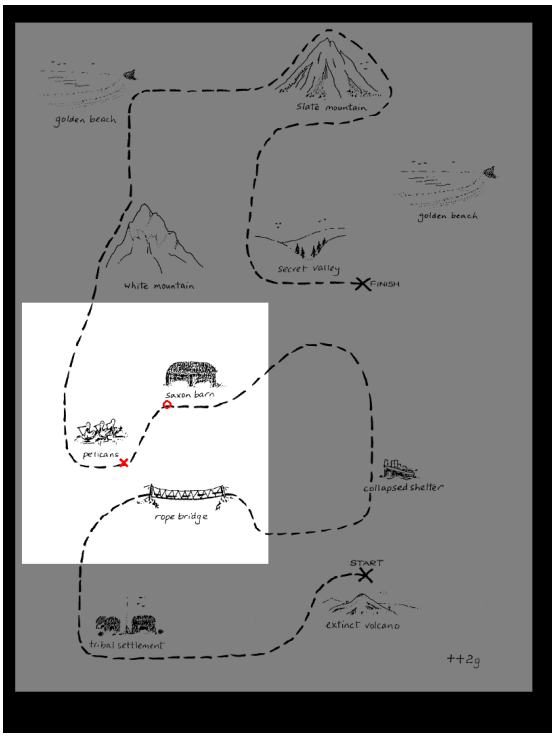


Figure 6.13: Example of a sub-route as displayed to a human judge. Each sub-route is marked with start and end points (○ and × in red).

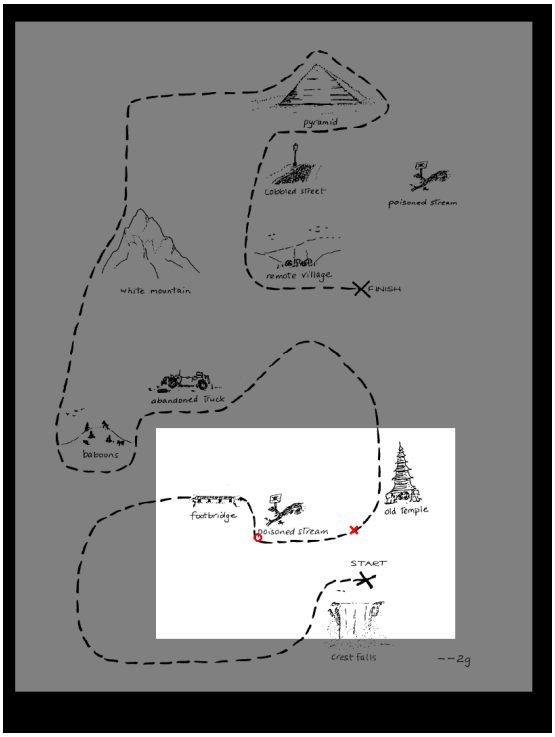


Figure 6.14: Example of a sub-route as displayed to a human judge. Each sub-route is marked with start and end points (○ and × in red).

Factor	S Sq	Df	F	Pr(>F)
Model	**4845.3**	3	783.9	<**0.001**
Sub-route	1140.0	93	5.95	<0.001
M:S	2208.7	279	3.84	<0.001
Residuals	3263.5	1584		

Table 6.6: Two-way ANOVA with factors model (4 possibilities), and sub-route (94 possibilities). Results show a significant model effect accounting for most of the variance. This means that the scores assigned to the navigational units by human judges are significantly influenced by the model used to generate the navigational units. Other effects are present, such as route effects and an interaction between the two factors.

Model Comparison	t value	Pr(> |t|)
Mismatch : Baseline	-16.974	<0.001
SSM : Baseline	23.882	<0.001
Real Giver : Baseline	23.192	<0.001
SSM : Mismatch	40.857	<0.001
Real Giver : Mismatch	40.507	<0.001
SSM : Real Giver	1.171	**0.646**

Table 6.7: Tukey HSD shows that all models are assigned significantly different scores by judges, apart from SSM and Real Giver. This asserts that, although only 33% of SSM navigational units match Real Giver navigational units (as shown in Table 6.4), the quality of the navigational units are not judged to be significantly different.

original data, against which accuracy was being judged in Table 6.4. This demonstrates that when many answers are feasible, scoring correctness against the original human navigational units is unsuitable. It also firmly demonstrates the suitability of our spatial representation, and the strength of the generative model we have induced for the task.

6.8 Conclusion

In this Chapter, we presented a complete generative probabilistic model goal-directed user behaviour in a spatial navigation domain. Specifically, we presented a model of Instruction Giver behaviour conditioned on spatial user goals and on the previous Follower behaviour. We applied this model to the Map Task dialog corpus, and to

our knowledge, this work constitutes the first to model the Instruction Giver as a goal-directed probabilistic generative model.

In spatial navigation domains, user goals take the form of spatial trajectories across a landscape. The disparity between these spatial routes, which are represented as raw pixel coordinates, and their grounding as semantic utterances creates an interesting challenge compared to conventional slot-filling domains. The variability in expressing the same underlying spatial goal using different navigational units is another important factor which we considered in our modelling. We showed how to represent spatial goals in a navigational domain, and validated our representation by inducing (fully from data) a generative model of the Giver's semantic utterances conditioned on the spatial goal and the previous Follower act. Specifically we demonstrated how abstracting away from spatial routes to a meaningful feature-based representation facilitates reasoning and admits generalisation to new routes not encountered at training time. The probabilistic formulation of our model allowed us to capture the range of plausible behaviour given the same underlying goal, a property frequently exhibited by human users in the domain.

We evaluated the model, first intrinsically, by measuring how well it predicts held-out data, compared to simpler forms of our model which lack an explicit model of the spatial user goals. We demonstrated substantial gains over the alternative models, using held-out probability and Perplexity. However, because the intrinsic metrics can be difficult to interpret especially in the absence of an upper bound, we conducted two task-based evaluations. The first is an accuracy-based metric, in line with approaches of user model evaluation in the literature, which we argue is an unsuitable metric as it assume a single correct answer and a range of incorrect ones. The second is a qualitative assessment of the model output using human judgement (collected over Mechanical Turk) which demonstrated that the quality of the output of our model does not significantly differ from the performance of humans on the same task.

The model we presented can be extended in a number of ways. We have thus far assumed that the spatial routes are observed, and were tasked with generating an appropriate description of the route. One extension is to model the route selection process, i.e. model which part of the route, and how much of the route, to describe at any given point in the dialog. Another extension is to replace the simple bigram-based interaction component in our mode with a more sophisticated conversational dynamics model, which takes more of the dialog into account, possibly as dialog features.

A direct application of this work is robot guidance, by using the model of the

Giver's behaviour which we have induced as a simulator to induce an optimal Follower (an MDP-based dialog manager that interprets and follows navigational instructions). Another variation would be to learn a generative model of the Follower, by extracting features from Follower maps (labelled with routes drawn by real Followers). Finally, this work, and in particular the spatial goal representation, has broader applications beyond simulation for example for systems that describe routes to users.

Chapter 7

Conclusion and Future Work

7.1 Contributions of this thesis

We started this thesis with the claim that generative probabilistic models offer us a flexible and mathematically principled way of modelling user behaviour in dialog. Specifically, we were interested in modelling the *goal-directed* user behaviour, exhibited in task-oriented dialogs, and in capturing the *variability* exhibited by users while being consistent with their goals. We have introduced models to allow us to improve upon existing approaches in well-studied problems in the first instance (informational, slot-filling dialog domains), and then to tackle a new problem domain in which user modelling for automated dialog management has not been attempted (spatial navigation). The specific contributions of this thesis can be summaries as follows:

1. We have demonstrated that generative probabilistic models offer a suitable framework for modelling goal-directed user behaviour in co-operative, task-oriented dialog domains. We achieved goal-directed behaviour by defining explicit variables for user goals, and conditioning user behaviour (at the level of semantic utterances) on these goals. We have demonstrated that variability (which users exhibit while being consistent with their goals) naturally arises from our probabilistic representation (Chapters 5 and 6).

2. We applied this framework to three standard dialog resources, the DARPA Communicator corpus (Chapter 5), the Let's Go corpus (Chapter 5), and the HCRC Map Task corpus (Chapter 6).

3. Previous stochastic user models which have been driven by real dialog data have only modelled observable behaviour and lacked an explicit model of user goals

(Georgila et al., 2005a, 2006). In contrast, we have proposed to explicitly model the user goals. As a result, we have shown that we are better able to capture user behaviour compared to these goal-free approaches (Sections 5.6 and 6.6).

4. We have shown how to derive a suitable representation for latent categorical user goals, which can have multiple surface realisations (e.g. New York, NYC) and can be corrupted by speech recognition errors (e.g. New York, Newark). The representation we chose is based on topic models (Section 5.3.3).

5. We have shown how to derive a suitable feature-based abstraction for spatial user goals, which as raw data are presented numerically as a series of pixel coordinates (Section 6.3.2).

6. We have shown how to probabilistically map the feature-based abstraction of spatial goals to the navigational units which describe them. We did this by defining a Multivariate Normal Distribution (MVN) over the feature vectors given a navigational unit (defining one such MVN per unit). This representation has allowed us to implicitly handle the variability in expressing the same underlying spatial goal as navigational units (Section 6.3.3).

7. We have shown how to incorporate the goal representations into a larger generative model which also accounted for simple conversational dynamics using Markov models (Sections 5.3 and 6.3).

In the literature, probabilistic, goal-directed user models have been deemed too complex to be automatically induced from data. They have been described as "trainable but rarely trained" (Pietquin, 2012). Instead, their distributions have been handcrafted.

8. In contrast to previous probabilistic goal-directed approaches, we have in this thesis argued that models of sufficient complexity can in fact be automatically induced from data without handcrafting and without prior access to the domains's vocabulary. We have demonstrated this on three different corpora (Chapters 5 and 6).

Traditionally, user models are evaluated with a dialog manager in the loop. This approach pre-supposes the existence of a dialog manager. However, the primary purpose for constructing user models in first place is to automate the discovery of dialog management policies, and thus, there is a circularity of argument in assuming the existence of functioning dialog managers. For this reason (and a few others stated in

Section 3.5.2) we have argued against such evaluation. As an alternative we have adopted the some of the prediction-based metrics proposed in the literature (presented in Section 3.5.1.2), and have additionally proposed new ways for evaluation.

9. We have demonstrated the suitability of some of the prediction-based evaluation metrics previously adopted in the literature for evaluating data-driven user simulators, namely held-out probability and perplexity (Sections 5.6 and 6.6).

10. We have defined the task of classifying consistent dialogs from inconsistent ones, and demonstrated the ability of our model to perform well in the task, and to outperform strong baselines (Section 5.7).

11. We have demonstrated the appropriateness and strength of the spatial component, which we have derived and automatically induced, through a qualitative human judgement task (Section 6.7.3)

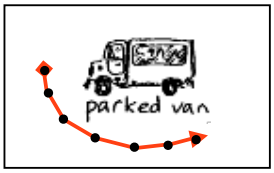
12. We have demonstrated the weakness of accuracy as a metric for evaluating user models, as it assumes one correct answer and heavily penalises a valid variability exhibited by models (Section 6.7).

Finally, to conclude this thesis, we now turn to some of the possible extensions to the work we have presented, followed by some of the most salient applications of our work.

7.2 Extensions to the models

A better model for ordinal slots

In Chapter 5 (Modelling Users with Latent Categorical Goals), we have presented a good model for categorical slots (e.g. place names) which can have multiple surface realisations (e.g. New York, NYC) and can be corrupted by speech recognition errors (e.g. New York, Newark). We have shown that topic models are suitable for capturing this ‘synonymy’ and ‘phonetic confusability’. However, we have not proposed a suitable representation for ordinal slots, such as dates and times, which should be treated differently, and have used a simple bigram model to model them. One good extension is to come up with a suitable representation for these kinds of slots.



$u_1 = \textsc{Position}(\textsc{Under}, \text{LM})$
$u_2 = \textsc{Move}(\textsc{Towards}, \textsc{Absolute\_East})$
$u_3 = \textsc{Move}(\textsc{To}, \textsc{Page\_Bottom})$

Figure 7.1: Each spatial route (illustrated with a red line) can be described using multiple different navigational units (e.g. u_1, u_2, or u_3). As raw data, a route is presented numerically as a series of pixel coordinates (marked in black dots), which we abstract away from using a feature-based representation. We then probabilistically map the features vectors, which we derive, to the navigational units which describe the route.

Modelling the route-selection process

In Chapter 6 (Modelling Users with Spatial Goals), we focussed on deriving an appropriate abstraction for the spatial routes, and probabilistically mapping this abstraction to the navigational units which describe them. See Figure 7.1 for an example of a spatial sub-route and some navigational units which can describe it. For the sake of simplification, we assumed that the routes are observed, and were tasked with producing an appropriate description of the route, as a navigational unit, in response to the follower. One appropriate extension is to model the process by which parts of the route are selected for description; that is, which set of points to talk about, whether to move on to a new set of points or re-describe the current set to facilitate better understanding.

Extending the interaction model

As part of our generative model, we defined an interaction model responsible for capturing the conversational dynamics between the two interlocutors taking part in the dialog. For example, in Chapter 5 (Modelling Users with Latent Categorical Goals), the semantic user utterance u at any given point was conditioned on the previous semantic machine utterance m.

$$p(u|m) \tag{7.1}$$

Similarly, in Chapter 6 (Modelling Users with Spatial Goals), the Giver's dialog act g was conditioned on the previous dialog act of the follower f.

$$p(g|f) \tag{7.2}$$

In both cases the conversational dynamics were modelled using a second-order Markov model. This is the simplest model that can be defined to capture the conversational dynamic, but can be extended to better account for the behaviour of the interlocutors. One way to do this is to model *features* of the dialog up the the point of the utterance of concern, and to condition the utterance on the features. For instance, in slot filling domains, we could condition on whether values for the different slots have been provided, how many times they have been provided, etc.

7.3 Applications of the models

Our primary motivation for modelling user behaviour was to build better dialog systems. More specifically, we advocated using the models as user simulators within a decision theoretic framework using sample-based algorithms which allow dialog managers to learn optimal policies of behaviour through interaction. We also motivated the idea of using the models as tools for exploratory analysis of user behaviour, and the possibility of exploring a different class of algorithms for policy discovery, namely model-based algorithms, which directly exploit distributions provided by our model to discover optimal policies of behaviour. In this section, we will discuss in more detail three possible future directions for the user models we have developed in this thesis.

Deploying the user models as simulators

We started this thesis by motivating modelling user behaviour for the purpose of simulation, and in particular, to train dialog managers within a decision theoretic framework through interaction with these simulators. The next natural step for our work is to begin to replace existing simulators with our generative models, and to examine questions such as: given better simulators (where better simulators are ones that more closely resemble user behaviour, as demonstrated by our evaluation), do we:

- converge to the same dialog management policies *faster*?
- learn dialog management policies which are better able to cope with new users?
- learn different, *better* policies? (although evaluation of dialog managers is also a difficult problem)

The first question addresses the tractability of the reinforcement learning algorithms typically deployed for dialog management policy discovery. By only considering rea-

sonable states, as provided by our user models, are we able to cut down on the policy search space?

Do better models lead to better dialog management policies?

The second and third questions address the quality of the resultant management policy. While this is typically evaluated through the interaction with simulators (another circularity), we are advocating evaluating dialog managers by interacting with real users, or in other words, through the task of dialog management itself. We would then be able to indicate whether our evaluation metrics correlate with metrics for evaluating the resultant management policy.

The question of whether better user models lead to better dialog management policies has been previously addressed in the literature. Ai et al. (2007) find that an MDP-based dialog manager trained on a 'random' simulator outperforms the same MDP trained on a probabilistic simulator which produces 'realistic' behaviour. However, this work does not evaluate how closely the simulators capture real user behaviour *prior to* using them to train MDP-based dialog manager. The one deemed 'random' might be sampling from the corpus of real user behaviour in a non-discriminating way, and thus producing more plausible training scenarios than their probabilistic (and crucially handcrafted) simulator, which might be biased; something which would come across if evaluated in the same fashion as in this thesis. Furthermore, this work evaluates the dialog manager's performance in terms of how certain the policy becomes after interacting with the different simulators, which tells us nothing about how good the policy is, or how well it would perform when interacting with real users.

We have instead in this thesis advocated evaluating the simulators in isolation from the MDP-based dialog managers they are intended to train (at least initially). The question of whether better simulators (i.e. those deemed better as a result of rigorous evaluation) produce better dialog managers (and whether the evaluation metrics correlate) then becomes an interesting line of questions. While the answers may be intuitive: objectively better simulators should produce better dialog managers, it is still a question worth exploring empirically.

User models as transition functions

Perhaps most promising of all is the notion that our user model can serve as the transition function in the MDP (which represents the dialog manager), thus enabling a

different range of techniques to be employed to learn dialog policies, namely planning methods (Section 2.3.4). Given that we have induced complete probability distributions of user behaviour, i.e. the range of possible user behaviour coupled with the associated probability of occurring, we have something qualitatively different to the ability to only produce *sample* of behaviour. Can we then directly exploit these distributions to discover optimal policies of behaviour, exactly (or at least, exact with respect to the induced distribution)?

Restricting the policy search space

A final idea involves thinking of optimal actions as subset of the plausible actions we've induced, which themselves are a subset of all possible actions. Consider how in Chapter 6 (Modelling Users with Spatial Goals) we induced a model of the plausible actions of instruction Givers, as a distribution (over 87 possible actions at any given point) with some actions being highly probable while others being highly improbable (most of the probability mass falls on a small subset of these actions). Can we use this distribution to inform our search for optimal Giver actions? Figure 7.2 illustrates how possible, plausible and optimal actions relate to one another. More concretely, in reinforcement learning, there is typically a trade-off between exploring new actions and exploiting actions which have been previously successful. Can this trial-and-error process be informed by the distributions over actions which we have already induced? Can our Giver models serve as a means of restricting the possible actions that can be taken, or at the very least, can they take priority in being explored? A similar idea has been applied in the dialog management literature (Henderson et al., 2008) and in the Robotics literature (Rosman and Ramamoorthy, 2012) which suggests that this can be a promising future direction.

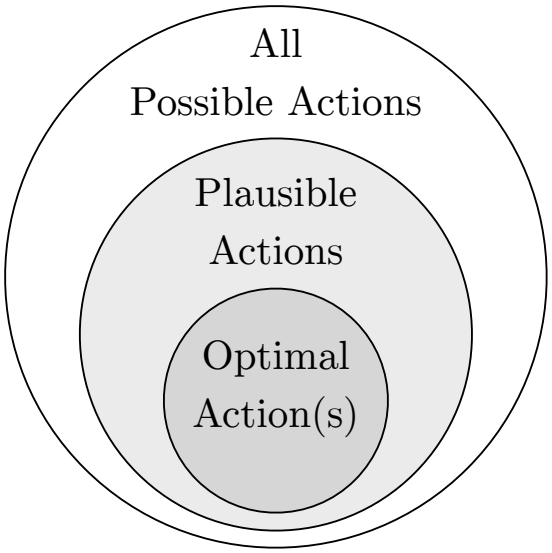


Figure 7.2: Our generative models we designed to learn the subset of *plausible* actions from all possible action. Of this subset of plausible actions is a smaller subset of optimal actions (or even just a single optimal action). One future extension involves thinking of the subset of plausible actions as a means of restricting the kinds of actions to consider at any given point (or restricting the policy's exploration to this subset).

Appendix A

Model Derivation

We show how to derive Equation 6.2 under Section 6.3. We start with the following Equation, where dependencies can be taken to indicate causal links between the variables:

$$p(g,u,f,W) = p(f)\ p(W)\ p(g|f)\ p(u|g,W) \tag{A.1}$$

This model says that f and W are root nodes, and that g depends only on f while u depends on g and W: a simple model based on the chronology of how the data may have been generated.

Because it is difficult to directly compute the distribution $p(u|g,W)$, we derive a different generative model which inverts the dependencies in Equation (A.1) making it easier to compute probabilities.

First, we use Bayes' theorem to substitute $p(g|f)$ with $\frac{p(f|g)\ p(g)}{p(f)}$:

$$p(g,u,f,W) = p(f)\ p(W)\ \frac{p(f|g)\ p(g)}{p(f)}\ p(u|g,W) \tag{A.2}$$

which allows the two terms, $p(f)$ and $\frac{1}{p(f)}$, to cancel out resulting in the following:

$$p(g,u,f,W) = p(W)\ p(f|g)\ p(g)\ p(u|g,W) \tag{A.3}$$

We then use Bayes' theorem to further substitute $p(u|g,W)$ with $\frac{p(g,W|u)\ p(u)}{p(g,W)}$:

$$p(g,u,f,W) = p(W)\ p(f|g)\ p(g)\ \frac{p(g,W|u)\ p(u)}{p(g,W)} \tag{A.4}$$

Because g and W are independent, we can susbtitute $p(g,W)$ with $p(g)P(W)$. we also make the assumption that g and W are conditionally independent given u, which allows us to rewrite $p(g,W|u)$ as $p(g|u)p(W|u)$:

$$p(g,u,f,W) = p(W)\ p(f|g)\ p(g)\ \frac{p(g|u)\ p(W|u)\ p(u)}{p(g)p(W)} \tag{A.5}$$

The following terms cancel out: $p(W)$ with $\frac{1}{p(W)}$ and $p(g)$ with $\frac{1}{p(g)}$, resulting in the following joint distribution:

$$p(g,u,f,W) = p(f|g)\ p(g|u)\ p(W|u)\ p(u) \tag{A.6}$$

The dependencies in the new model, described in Equation (A.6), no longer indicate causal links since they do not preserve the chronology of how the data may have been produced, but the new model does provide a convenient way of defining computing probabilities for our task.

Bibliography

Ai, H. and Litman, D. J. (2008). Assessing dialog system user simulation evaluation measures using human judges. In *Proceedings of ACL-08: HLT*.

Ai, H., Tetreault, J. R., and Litman, D. J. (2007). Comparing user simulation models for dialog strategy learning. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anderson, A. H., Bader, M., Bard, E. G., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H. S., and Weinert, R. (1991). The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

Asher, N., Lascarides, A., Lemon, O., Guhe, M., Rieser, V., Muller, P., Afantenos, S., Benamara, F., Vieu, L., Denis, P., et al. (2012). Modelling strategic conversation: the STAC project. In *The 16th workshop on the semantics and Pragmatics of Dialogue (SeineDial12). Paris*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Black, A. W. and Eskenazi, M. (2009). The Spoken Dialogue Challenge. In *Proceedings of SIGDial 2009*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boye, J. (2007). Dialogue management for automatic troubleshooting and other problem-solving applications. In *Proceedings of SIGDial Workshop on Discourse and Dialogue*, Association for Computational Linguistics, pages 247–255.

Chandramohan, S., Geist, M., and Pietquin, O. (2010). Sparse approximate dynamic programming for dialog management. In *Proceedings of SIGDial*, pages 107–115. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Chung, G. (2004). Developing a flexible spoken dialog system using simulation. In *Proceedings of ACL*, page 63. Association for Computational Linguistics.

Cuayahuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-computer dialogue simulation using hidden Markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005*, pages 290–295.

Daubigney, L., Geist, M., and Pietquin, O. (2012). Off-policy learning in large-scale POMDP-based dialogue systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012*, pages 4989–4992.

Denecke, M., Dohsaka, K., and Nakano, M. (2005). Fast reinforcement learning of dialogue policies using stable function approximation. In *Natural Language Processing IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User modeling for spoken dialogue system evaluation. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 80–87.

Elsner, M. and Charniak, E. (2008). You talking to me? a corpus and algorithm for conversation disentanglement. In *ACL*, pages 834–842.

Eshky, A., Allison, B., Ramamoorthy, S., and Steedman, M. (2014). A generative model for user simulation in a spatial navigation domain. In *Proceedings of the 2014 Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden. Association for Computational Linguistics.

Eshky, A., Allison, B., and Steedman, M. (2012). Generative goal-driven user simulation for dialog management. In *EMNLP-CoNLL 2012*, pages 71–81, Jeju Island, Korea.

Gasic, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. (2013). POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the SIGDIAL 2013 Conference*, pages 214–222, Metz, France. Association for Computational Linguistics.

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

Georgila, K., Henderson, J., and Lemon, O. (2005a). Learning user simulations for information state update dialogue systems. In *InterSpeech*, pages 893–896.

Georgila, K., Henderson, J., and Lemon, O. (2006). User simulation for spoken dialogue systems: learning and evaluation. In *InterSpeech*.

Georgila, K., Lemon, O., and Henderson, J. (2005b). Automatic annotation of communicator dialogue data for learning dialogue strategies and user simulations. In *SemDial 2005*.

Giuliani, M., Petrick, R. P. A., Foster, M. E., Gaschler, A., Isard, A., Pateraki, M., and Sigalas, M. (2013). Comparing task-based and socially intelligent behaviour in a robot bartender. In *Proceedings of the ACM International Conference on Multimodal Interaction (ICMI 2013)*, Sydney, Australia.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *PNAS*, 101(suppl. 1):5228–5235.

Henderson, J. and Lemon, O. (2008). Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 73–76. Association for Computational Linguistics.

Henderson, J., Lemon, O., and Georgila, K. (2005). Hybrid reinforcement/supervised learning for dialogue policies from Communicator data. In *IJCAI workshop on knowledge and reasoning in practical dialogue systems*, pages 68–75.

Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

Janarthanam, S. and Lemon, O. (2009). A two-tier user simulation model for reinforcement learning of adaptive referring expression generation policies. In *Proceedings of SIGDial 2009*, pages 120–123.

Janarthanam, S., Lemon, O., Bartie, P., Dalmas, T., Dickinson, A., Liu, X., Mackaness, W., and Webber, B. (2013). Evaluating a city exploration dialogue system with integrated question-answering and pedestrian navigation. In *ACL*.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P. (2002). MATCH: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics.

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

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *CoRR*, cs.AI/9605103.

Keizer, S., Gašić, M., Jurčíček, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010). Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDial 2010*.

Lemon, O., Georgila, K., and Henderson, J. (2006). Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *IEEE Spoken Language Technology Workshop, 2006.*, pages 178–181.

Levin, E., Narayanan, S., Pieraccini, R., Biatov, K., Bocchieri, E., Fabbrizio, G. D., Eckert, W., Lee, S., Pokrovsky, A., Rahim, M., Ruscitti, P., and Walker, M. (2000a). The AT&T-DARPA communicator mixed-initiative spoken dialog system. In *ICSLP*.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using Markov decision process for learning dialogue strategies. In *Proc. ICASSP*, pages 201–204.

Levin, E., Pieraccini, R., and Eckert, W. (2000b). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Levit, M. and Roy, D. (2007). Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 37(3):667–679.

Li, C., Xu, B., Wang, X., Ge, W., and Hao, H. (2013). Simulated spoken dialogue system based on IOHMM with user history. In Zhou, G., Li, J., Zhao, D., and Feng, Y., editors, *Natural Language Processing and Chinese Computing*, volume 400 of *Communications in Computer and Information Science*, pages 83–92. Springer Berlin Heidelberg.

Li, L., Williams, J. D., and Balakrishnan, S. (2009). Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *InterSpeech*, pages 2475–2478.

Lison, P. (2010). Towards relational POMDPs for adaptive dialogue management. In *Proceedings of the ACL 2010 Student Research Workshop*, ACLstudent '10, pages 7–12, Stroudsburg, PA, USA.

Lison, P. (2013). Model-based Bayesian reinforcement learning for dialogue management. In *InterSpeech 2013*.

Neal, R. (1991). Bayesian Mixture Modeling by Monte Carlo Simulation. Technical report, University of Toronto.

Parent, G. and Eskenazi, M. (2010). Toward better crowdsourced transcription: Transcription of a year of the Let's Go bus information system data. In *IEEE Workshop on Spoken Language Technology, 2010. SLT 2010*, pages 312–317.

Pietquin, O. (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*. PhD thesis, Faculté Polytechnique de Mons, TCTS Lab (Belgique).

Pietquin, O. (2012). Statistical user simulation for spoken dialogue systems: what for, which data, which future? In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 9–10. Association for Computational Linguistics.

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

Pietquin, O., Geist, M., and Chandramohan, S. (2011a). Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three*, IJCAI'11, pages 1878–1883. AAAI Press.

Pietquin, O., Geist, M., Chandramohan, S., and Frezza-Buet, H. (2011b). Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Trans. Speech Lang. Process.*, 7(3):7:1–7:21.

Pietquin, O. and Hastie, H. (2013). A survey on metrics for the evaluation of user simulations. *Knowledge Eng. Review*, 28(1):59–73.

Reitter, D. and Moore, J. D. (2007a). Predicting success in dialogue. In *ACL*.

Reitter, D. and Moore, J. D. (2007b). Successful dialogue requires syntactic alignment. Poster at 20th Annual CUNY Conference on Human Sentence Processing.

Rieser, V. and Lemon, O. (2008). Learning effective multimodal dialogue strategies from wizard-of-oz data: Bootstrapping and evaluation. In *ACL*, pages 638–646.

Rosman, B. and Ramamoorthy, S. (2012). What good are actions? Accelerating learning using learned action priors. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6. IEEE.

Rossignol, S., Pietquin, O., and Ianotto, M. (2011). Training a BN-based user model for dialogue simulation with missing data. In *IJCNLP*, pages 598–604.

Roy, N., Pineau, J., and Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of ACL 2000*, ACL '00, pages 93–100, Stroudsburg, PA, USA.

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

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. (2007a). Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *HLT-NAACL (Short Papers)*, NAACL-Short '07.

Schatzmann, J., Thomson, B., and Young, S. (2007b). Statistical user simulation with a hidden agenda. In *Proceedings 8th SIGDial Workshop on Discourse and Dialogue*.

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

Schatzmann, J. and Young, S. (2009). The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT 2002*.

Singh, S., Litman, D. J., Kearns, M., and Walker, M. A. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. (2000). Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*.

Strik, H., Russel, A., Heuvel, H., Cucchiarini, C., and Boves, L. (1997). A spoken dialog system for the Dutch public transport information service. *International Journal of Speech Technology*, 2(2):121–131.

Suendermann, D., Evanini, K., Liscombe, J., Hunter, P., Dayanidhi, K., and Pieraccini, R. (2009). From rule-based to statistical grammars: Continuous improvement of large-scale spoken dialog systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.*, pages 4713–4716.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Thompson, H. S., Anderson, A., Bard, E. G., Sneddon, G. D., Newlands, A., and Sotillo, C. (1993). The HCRC Map Task corpus: natural dialogue for speech recognition. In *Proceedings of the workshop on Human Language Technology*, HLT '93, pages 25–30, Stroudsburg, PA, USA.

Thomson, B., Schatzmann, J., and Young, S. (2008). Bayesian update of dialogue state for robust dialogue systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008.*, pages 4937–4940.

Vogel, A. and Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of ACL*, pages 806–814. Association for Computational Linguistics.

Walker, M., Litman, D., Kamm, C., and Abella, A. (1998a). Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech and Language*, 12(4):317 – 347.

Walker, M. A., Fromer, J. C., and Narayanan, S. (1998b). Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, pages 1345–1351. Association for Computational Linguistics.

Williams, J. D. (2007). 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*, pages 1–8. Association for Computational Linguistics.

Williams, J. D. (2008). Evaluating user simulations with the Cramer-von Mises divergence. *Speech Communication*, 50(10):829–846.

Williams, J. D. (2012). A critical analysis of two statistical spoken dialog systems in public use. In *IEEE Spoken Language Technology Workshop (SLT), 2012*, pages 55–60. IEEE.

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

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

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Young, S. J. (2000). Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.