

A Probabilistic Model of Human/Computer Dialogue
with Application to a
Partially Observable Markov Decision Process

First-year report

Jason D. Williams
Churchill College

Supervisor: Prof. Steve Young



Machine Intelligence Laboratory
Department of Engineering
University of Cambridge

29 August 2003

A Probabilistic Model of Human/Computer Dialogue with Application to a Partially Observable Markov Decision Process

First-year report

Jason D. Williams

Abstract

While numerous models of human/human conversation have been proposed and evaluated, human/computer conversation currently lacks a model which accounts well for the increased uncertainty introduced by the automated speech recognition (ASR) channel. Further, while recent research has shown the feasibility of modelling the sources of uncertainty in human/computer conversation explicitly, no work has constructed a comprehensive model informed by interaction data to inform optimal machine behaviour. This work proposes to address these research opportunities with three endeavours. First, a series of human/human conversations will be conducted in which human subjects communicate through an ASR-like channel. Second, based on these conversations, a probabilistic model of dialogue focused on grounding and turn-taking will be constructed and evaluated. Lastly, a user model will be developed and integrated with the dialogue model; this combined model will then be applied to a Partially-Observable Markov Decision Process to find optimal machine behaviours. Feasibility of each phase is discussed citing other research in the field and original work. Finally, a research schedule is proposed.

The work discussed in this report was carried out between October 2002 and August 2003, and proposes a schedule of work from September 2003 to August 2005. This report contains approximately 15,000 words.

Table of contents

Chapter 1: Introduction	1
Chapter 2: Human/human and human/computer conversation	3
2.1 <i>A brief characterisation of human/human conversation</i>	3
Formal theories: speech acts, plans, and mutual beliefs	3
Conversational analysis: grounding and turn-taking.....	4
Summary.....	4
2.2 <i>The human/computer conversational interface</i>	4
Feature extraction and speech recognition.....	4
Utterance segmentation and representation	5
Parsing	5
Dialogue state, system actions, and dialogue management.....	6
Summary.....	6
2.3 <i>Observed characteristics of human/computer conversation</i>	7
Turn-taking, turn size, and initiative	7
Grounding: backchannel and confirmations	7
Recognition errors: user reactions, corrections, and confirmation strategies.....	8
Experience and priming	9
Summary.....	10
Chapter 3: Current approaches	11
3.1 <i>Traditional methods</i>	11
3.2 <i>Better confidence measures</i>	12
3.3 <i>Localised goal and dialogue act detection</i>	13
3.4 <i>Ranked state hypotheses</i>	13
3.5 <i>Decision-theoretic methods</i>	14
3.6 <i>Markov Decision Processes (MDPs)</i>	15
Introduction to Markov Decision Processes	15
Overview of SDS/MDP literature	16
Training issues	17
State space and action set selection.....	17
Reward functions	18
3.7 <i>Partially Observable Markov Decision Processes (POMDPs)</i>	19
Introduction to Partially Observable Markov Decision Processes	19
Augmented MDPs.....	20
POMDPs with approximate solutions.....	20
3.8 <i>Summary of current approaches</i>	22
Chapter 4: Project proposal	25
4.1 <i>Project proposal, part 1: Data collection</i>	25
Experimental design	25
Theoretical design considerations.....	27
Practical design considerations	27
4.2 <i>Project proposal, part 2: Dialogue model creation</i>	28
Data preparation.....	28
Data preparation: Toy example.....	29

	Iterative dialogue model development	30
	Dialogue model development: Toy example	30
	Model evaluation	31
4.3	<i>Project proposal, part 3: Dialogue management</i>	32
	Creation of user model	32
	Creation of user model: Toy example	32
	Specification of reward values	33
	Specification of reward values: Toy POMDP example	35
	POMDP assembly: Toy POMDP example	35
	Solution methods	36
	Extensions to the Toy POMDP example	38
4.4	<i>Project proposal, part 4: Optional extension</i>	38
4.5	<i>Project criticisms and risks</i>	38
	Chapter 5: Project schedule	41
	Chapter 6: References	43

Chapter 1: Introduction

Conversation is characterised by intelligent behaviour under uncertainty. Viewed broadly, Speaker A's uncertainty arises from four sources:

1. Speaker A is uncertain about the beliefs or goals of Speaker B
2. Speaker A is uncertain about what Speaker B actually said (or if Speaker B was speaking to Speaker A)
3. Speaker A is uncertain about the current state of the conversation
4. Speaker A is uncertain about the effects his speech on Speaker B

While humans generally manage the uncertainty in a conversation well, spoken dialogue systems (SDSs) have struggled. Recently, researchers have begun modelling these uncertainties explicitly. While recent studies have shown progress, the field currently lacks several vital insights:

- *Dialogue model*: While much work has been done on establishing good descriptive and computational models of grounding and turn-taking in human/human conversation, it is unclear whether these models are appropriate in the presence of an automated speech recognition (ASR) channel. In particular, we lack an accepted enumeration of the state space and action set for the human and computer.
- *Dialogue manager*: In order for dialogue managers to reason about future states of the conversation, they need to know both the current state of the conversation and how their conversational partner is likely to respond – i.e., a *user model*. Yet to date, no SDSs have combined an approach which explicitly models all forms of dialogue uncertainty with a data-derived user model.

This paper proposes three research endeavours. First, a study of human/human conversation in the presence of an ASR-like channel is proposed. The second endeavour constructs a computational model of these dialogues, focused on grounding and turn-taking. Finally, the third endeavour constructs a user model and seeks methods to enable a machine to communicate effectively given this model.

This paper proceeds as follows. Section 2 provides background on the differences between human/human and human/computer dialogue, with an emphasis on the effects of the ASR channel. Section 3 reviews current approaches to handling uncertainty in SDSs.

Section 4 proposes a study to address current shortcomings and discusses its feasibility. Section 5 outlines a project schedule.

Chapter 2: Human/human and human/computer conversation

This section first gives an overview of phenomena present in human/human (HH) conversation. Next, human/computer (HC) conversation is characterised, focusing on the properties of the ASR channel. Finally, differences between HH and HC conversation are discussed.

2.1 A brief characterisation of human/human conversation

HH conversation has been studied and described in its own right for some time. Researchers have proposed many competing models, annotation taxonomies, and theoretical principles for virtually all observed phenomena: even an introductory presentation is beyond the scope of this work. This sub-section gives a selective account, focusing on relevant topics for this work.

Formal theories: speech acts, plans, and mutual beliefs

Researchers have devised many hierarchical levels of description for speakers' actions in conversation. (Austin, 1962) calls these actions *speech acts*, and distinguishes between three high-level categories: locutionary acts, illocutionary acts, and perlocutionary acts.

A *locutionary act*, the lowest level of description, is the act of producing an understandable utterance which puts together recognisable phonemes into recognisable words according to the grammar or conventions of a language, and which has some particular sense and meaning. An *illocutionary act*, a higher level of description, is the act performed *in* saying something, such as asking a question, giving some information etc. Illocutionary acts consist of an *illocutionary force*, specifying the type of action (e.g., suggesting, requesting, informing), and *propositional content*, specifying the details of the action, (e.g., Boston or three PM). A *perlocutionary act*, the highest level of description, is the *effect* of the utterance in the broader context – for example, persuasion or surprise.

Austin's work proposes classes of illocutionary acts. (Searle, 1976) improves on these, positing categories of illocutionary acts familiar to researchers today: *representatives*, *directives*, *commissives*, and *expressives*.

(Cohen and Perrault, 1979) contribute the notion of speech acts as *plan operators* which affect the beliefs of the speaker and hearer. Starting with illocutionary acts, they add (1) a planning system (i.e., a formal language for describing states and events in the world, including people's beliefs), and (2) a definition of how speech acts change the state of the world and speakers' separate beliefs and mutual beliefs. This system allows a listener to infer a speaker's beliefs through logical methods.

Conversational analysis: grounding and turn-taking

Conversational Analysis examines actual conversations and seeks to inductively derive theories of dialogue. Whereas formal theories have given rise to the useful notion of mutual belief and insights into planning, Conversation Analysis has contributed grounding and turn-taking.

First, note that formal theories have assumed complete and flawless understanding between speakers. In practice misunderstandings occur, and researchers have found that speakers engage in an extensive collaborative, coordinated series of exchanges used when instantiating a new mutual belief, perhaps to reduce communication errors. (Clark and Brennan, 1990) explains this process as making contributions to the common ground of a conversation, a process called *grounding*. (Traum, 1994) adds a finite-state description of the grounding process, and proposes methods for choosing between available conversational actions.

Second, researchers have quantitatively analysed several aspects of conversation. Although conversants usually speak one at time, there are frequent but brief periods of overlap in which more than one conversant is speaking. These periods consume less than 5% of the speech stream (Levinson, 1983). Thus conversations are usually divided into “turns” in which one speaker maintains the “floor” in a conversation. (Flammia, 1998) annotates 100 hours of conversations between customers and agents. He finds that over 80% of user utterances contain fewer than 12 words, with over half being 4 words or less. Nearly half of the customers’ dialogue turns were acknowledgements, contributing to the grounding process. Flammia also finds that higher-level conversation can be described with stack-like operators, but notes that stack depth remains shallow.

Summary

In goal-directed conversations, speakers generally understand most all words they exchange. While the grounding process consumes much of the conversation, corrections do not. In addition, conversants’ exchanges are virtually instantaneous – enough so that they often overlap.

2.2 The human/computer conversational interface

This section strives to describe the operating conditions of the HC interface. Each of the components described below could occupy a textbook on its own: this description is necessarily selective and broad.¹

In this work we are concerned with goal-directed HC conversation. Most research in SDSs has focused on this area, and we implicitly limit ourselves to this area of inquiry.

Feature extraction and speech recognition

Speech arrives at the feature extractor as a discrete, sampled audio signal. A *feature extractor* typically looks at a *sample window* of some milliseconds to produce a *feature vector*. Most feature sets are designed to remove pitch baseline and variability, and attempt to capture prosodically-independent information relevant for word recognition.

¹ Interested readers are referred to (Glass, 1999) and (Zue, 1997) for overviews and further references.

Features vectors are passed to the *speech recogniser*. The speech recogniser employs *acoustic models* to map sound to phonemes, a *lexicon* to map phonemes to words, and a *language model* to map words to utterances. Most recognisers model speech using *Hidden Markov Models*, in part to model (and remove) variations in speaker speed. Language models may either be *rule-based grammars* which compactly specify a possibly infinite, related set of recognisable utterances, or *N-gram* models which estimate local probabilities of word sequences. Speech recognisers typically consider a combined probability at the phoneme, word, and language model level when decoding, although some approximation (often in the form of a search *beam width*) is required to retain computational tractability.

Utterance segmentation and representation

The user's speech is usually segmented into *user utterances*, accomplished using either an *end-pointer* or *silence models*.

An end-pointer, which resides between the feature extractor and the speech recogniser, determines where an utterance begins and ends through acoustic analysis of the speech signal *without performing recognition*. Once an end-pointer has identified an utterance, it submits the utterance to the speech recogniser. The end-pointer may either ignore speech detected while the recogniser is working or queue additional speech for subsequent processing.

Alternately, systems may add a *silence model* to the phoneme set, lexicon, or language model. The speech recogniser performs continuous recognition, explicitly recognising silence. Speech may then be segmented into utterances by taking all speech not separated by some number of silence observations.

Commercial telephony systems usually employ an end-pointer, reducing speech recognition load and hardware requirements. Silence models are sometimes preferred in research systems as they provide more information on which to base turn-taking decisions.

After an utterance has been segmented and recognised, the speech recogniser produces some representation of the speech – typically in the form of a *word string*, an *N-best list* of word strings, or a *word lattice*. Additional information such as an *acoustic confidence score* or *utterance duration* may also be supplied.

Parsing

The speech recogniser's result is submitted to a *parser*, which extracts meaningful content from the result.² This content can be expressed as *concepts* (e.g., from=London, to=Boston), *goals* (e.g., “plan-flight”, “check-flight-status”), and/or *dialogue acts* (e.g., “wh-question”, “inform”, “affirmative”). The parser may also return additional information such as a *parse confidence score*, or a *coverage score*.

² For rule-based language models, parsing and recognizing may be collapsed to one step.

Dialogue state, system actions, and dialogue management

The parser's results are used to update the current *dialogue state*. There are a host of techniques for modelling dialogue state, including *finite state automata*, *frames*, *information states*, and *belief states*. These methods are investigated in detail in Chapter 3.

Based on the dialogue state, the *dialogue manager* determines an action for the system to take. These actions can include *dialogue actions* (producing utterances, waiting), *recognition actions*³ (selecting a grammar or parser), and/or *programmatic actions* (requesting information from a database, submitting information to a database).

Non-silent dialogue actions produce a (usually immediate) *system utterance*, often through *text-to-speech* in research systems and *pre-recorded prompts* in commercial systems.

While it is producing a system utterance, the system may monitor the input channel for user speech; if detected, the system may stop playback of the system utterance. This functionality is called *barge-in*, and can be implemented using either an end-pointer or silence models.

Most SDSs function synchronously, assuming user and system turns alternate.⁴ The system waits for the end of a user utterance, examines it, takes zero or more database actions, possibly taking a recognition action, and ultimately producing one or more dialogue actions. This sequence of events invariably results in a pause between the end of a user utterance and the beginning of a system utterance; this pause can either be left silent, or filled with a “system-busy” sound.

Summary

In this work we focus on the affects of the *ASR channel*, which consists of all components between the user and the dialogue model – i.e., the feature extractor, speech recogniser, parser, and utterance segmentation components. The ASR channel provides a SDS with (often noisy) observations about the (real) world.

Corruption in the ASR channel derives from several sources. Most crucially, the speech recogniser does not accurately report the words the user speaks, nor their prosody: utterances are subject to word *substitutions*, *deletions*, and *insertions*, as well as the removal of *meaningful prosodic information*.⁵ Moreover, both barge-in and the utterance segmentation process give rise to *turn-taking errors*. Even though the division of user's speech into turns and utterances is justified given evidence from Conversational Analysis, the division is an approximation and introduces *input noise* (in how the turns are segmented) and *input lags* (because of the processing delays). In addition, barge-in

³ Whereas acoustic models and the lexicon are typically fixed in a dialogue, the language model may be varied. For example, commercial systems usually use rule-based grammars which are selected based on the current dialogue state. Alternatively, a parser may be adjusted, for example in anticipation of an answer to a yes/no question instead of a wh-question.

⁴ Systems often employ a *silence timeout* to signify an empty user turn.

⁵ The system's utterances usually also have emaciated prosody, though with perhaps less severe consequences.

behaviour is not perfect; by stopping a prompt at the wrong time or failing to stop at the right time *barge-in errors* are introduced.⁶

2.3 Observed characteristics of human/computer conversation

Researchers have long understood that HH and HC conversation is very different. This section strives to characterise the traits of observed HC dialogues, drawing contrasts with HH dialogues where possible. Describing HC dialogue is challenging because findings are system dependent. The findings in this section are taken from systems viewed as broadly successful and representative of common practices in the field.

Turn-taking, turn size, and initiative

(Moore and Browning, 1992) describes parallel (i.e., same-domain, same-task) studies of HH and simulated HC (Wizard-of-Oz) dialogues. In these studies, the wizard has “perfect hearing” – i.e., no misrecognitions were simulated. The authors noted that no callers gave an indication they doubted the wizard was actually a computer. Users on average took 7.5 turns/call for the HC dialogues and 26.5 turns/call for the HH dialogues; users’ turns were longer on average for the HH dialogues: 5.0 words/turn for the HC dialogues vs. 7.0 words/turn for the HH dialogues.

(Doran et al., 2001) studies a variety of phenomena in goal-directed HC and HH dialogues. The authors annotate a corpus of HH dialogues and (real) HC dialogues from the same domain (airline reservations in DARPA Communicator) using the same tag set. The authors choose dialogue transcripts at random from 4 different Communicator sites (i.e., 4 different SDSs) to annotate. In the HH dialogues, both the expert (i.e., the travel agent or computer, as appropriate) and user display an average of 1.3 dialogue acts per turn, whereas in HC dialogues, the system displayed an average of 1.6 and the user 1.0 dialogue acts per turn.

The authors also find that the computer experts are more verbose than human experts: HH dialogues see on average 10.1 words/turn for experts and 7.2 words/turn for users, whereas HC dialogues see 17-33 words/turn for experts and 2.8-4.8 words/turn for users. Further, they annotate “initiative” at the end of each turn, and find that the expert retains initiative in about 90% of turns in HC dialogues vs. 50% in HH dialogues.

Thus there is consistent evidence that people use fewer words per turn in HC dialogues than in HH dialogues. This difference appears to arise from users’ expressing fewer dialogue acts in their turns, which may arise from systems’ retaining initiative more often.

Grounding: backchannel and confirmations

(Doran et al., 2001) also studies the frequency of dialogue acts in HH and HC dialogues. Counting both conversants, the top five dialogue acts in HH dialogues are (in order):

⁶ (Glass, 1999) notes that “in addition to the problem of being able to reliably detect barge-in... it becomes necessary to properly update the dialogue status to reflect the fact that barge-in occurred. The system must take into account where the interruption occurred during its response.” While some empirical results have shown that (lack of) barge-in is a significant contributor to user success (Litman et al., 1998), few (if any) systems incorporate location of barge-in into a dialogue or turn-taking model.

GiveTaskInfo, Acknowledge (back-channel), RequestTaskInfo, VerifyTaskInfo, and Affirm. The top five dialogues acts in HC dialogues are (in order): GiveTaskInfo, RequestTaskInfo, Offer, VerifyTaskInfo, and Apology.⁷ The pervasiveness of Acknowledge acts in HH dialogue is consistent with (Flammia, 1998), yet Acknowledge acts are virtually absent from HC dialogues. The higher ranking of VerifyTaskInfo in HC dialogues is consistent with the need for more confirmations. The higher ranking of Offer acts in HC dialogues (primarily from the system expert) is also consistent with the system's share of initiative.

Experts in HH dialogues use slightly more confirmations than users (247 for experts vs. 173 for users), but both parties use more short than long confirmations (340 vs. 80). By contrast, in HC dialogues, systems use both types of confirmation far more than users do (246 vs. 8). Moreover, systems use long confirmations about five times more often (210 vs. 36) than they used short confirmations.

Synthesising these observations, we see a radically different model of grounding emerging in HC dialogues: little to no backchannel, longer confirmations, more system-initiated confirmations, and virtually no user-initiated confirmations.

Recognition errors: user reactions, corrections, and confirmation strategies

Recognition errors are perhaps the most salient difference between HC and HH dialogues and have been studied in detail by a variety of researchers.

First, it is important to note that user speech in HC dialogues is often “clearer” than in HH dialogues: (Moore and Browning, 1992) report that 18% of caller turns in HC dialogues include “uhms” and false starts, compared to 40% of HH dialogues. This is unsurprising given that users' utterances are shorter in HC dialogues, and shorter utterances show fewer false-starts and self-corrections than longer ones.

Of course, misrecognitions still occur. (Doran et al., 2001) notes that “errors and misunderstandings are more frequent in the HC data, resulting in (some fairly verbose) efforts by the system to identify the problem and get the conversation back on track.”

Users' reactions to system errors are well-documented. (Priker and Loderer, 1999) simulated recognition errors with a WoZ study and found users tended to speak louder, slower, and with more internal pauses when correcting the system or in response to a rejection (e.g., “Sorry, I didn't understand.”). Responses to a rejection had a slightly increased pitch; corrections had a significantly raised pitch and increased volume on the item being corrected. Likewise, (Levow, 1998) finds corrections of recognition errors exhibited significantly heightened pitch variability and greater increases in duration, while corrections of rejection errors showed less pitch variability and fewer increases in duration. As a demonstration, the author shows how supervised learning can be used to classify these two correction types with 70-75% accuracy using prosodic features.

(Swerts et al., 2000) finds that users' correction-utterances which are *perceptually* indistinguishable from non-correction utterances (as determined by human judges) are

⁷ In addition, (Moore and Browning, 1992) report that users are more polite in the HC dialogues than in the HH dialogues: 31% of user turns in HC dialogues included an expression of politeness vs. 10 % of turns in HH dialogues.

analytically different in prosody. The authors find that corrections are more likely to be misrecognised by the system, but are no more or less likely to be rejected. Further, a positive correlation is found among three factors: (1) the distance between the introduction of a concept and where an error in understanding is detected by the user; (2) the difference in prosody exhibited by the user in correcting it; and (3) (unfortunately!) the likelihood it will be misrecognised.⁸

(Litman et al., 2000) applies these findings to improve system prediction of misrecognitions, showing how prosodic features can significantly improve prediction of ASR misrecognitions over traditional ASR-confidence (such as log-likelihood based confidence measures) alone.

(Shin et al., 2002) compares the relative success rates of different confirmation strategies in HC dialogues. The authors find that users discovering errors through implicit confirmations are less likely to get back on track, *and* take a *longer time* to get back on track than errors detected through rejections. Further successful error-recovery strategies employed by the user included: rephrasing and terminating error episodes (i.e., cancelling and starting over); attempting to repair long-running misunderstandings; and avoiding the use of contradictions.

Combining these insights, we see that techniques which humans use to resolve misunderstandings (e.g., prosodically accentuating a misunderstood word) actually cause *further* misunderstanding by the computer. This phenomenon could be due to the training data used for the acoustic models & feature extraction, which may not be trained on speech which includes corrections. Given current techniques and ASR technology, successful dialogue strategies strive to actively eliminate misrecognitions early and frequently, and make use of cancel-and-startover initiatives. Going forward, exploring how to integrate prosodic information into confidence measures at higher levels and into the parser appear to be promising avenues for research.

Experience and priming

Over time users' vocabulary converges to a subset which increases task completion. (Levow, 2003) demonstrates that novice users, adapt their behaviour to achieve significant reductions in rejected and falsely-accepted utterances. The author observes that a single user's vocabulary converges to a subset of the system's vocabulary as that user gains experience; however, different users' vocabularies show little overlap: in general only 50% of each user's reduced vocabulary is shared with any other user.

Finally, (Doran et al., 2001) observes that users change behaviour as they make more calls to the system, increasingly responding with more unsolicited information. In early calls, users average 1 unit of unsolicited information per call; this increases to 4 units by call 7. Moreover, callers learn how to become more successful – the unsolicited information provided in the first call is usually not successfully captured by the system; by call 7, 75% of unsolicited information has been successfully communicated to the system.

⁸ The authors also note that systems that produce longer tasks but fewer misrecognitions and subsequent corrections are preferred by users, discussed further in Section 0.

Despite the limitations of SDSs, there is strong evidence that callers adapt their behaviour to become more successful. Designers have capitalised on this effect in an effort to boost system performance by *priming* callers with examples of what they can say: (Sheeder and Balough, 2003) provides a compelling case study.

Summary

The differences observed between HC conversation and HH conversation are striking. Symmetries evident in HH dialogues for initiative, turn length, and density of dialogue-acts per turn become markedly asymmetric in HC dialogues. A radically different model of grounding emerges in which frequent back-channel is replaced with frequent repetition of content. When recognition errors go undetected for longer periods of time, they become increasingly problematic to resolve. Finally, callers adapt their behaviour to become more successful, but different callers converge to different behaviours.

These aspects of HC dialogue also account for the majority of problems in real systems. In a quantitative survey of the sources of failed interactions with commercial speech recognition services, (Balantine, 2003) cites *turn-taking problems* and *falsely-accepted user utterances* as the two most common root causes – both intimately linked with the observations presented in this section.

While the grounding and turn-taking process in HH dialogue has been well described through computational models, HC dialogues differ enough in both operational characteristics and practical observation to suggest that HH models of grounding and turn-taking are not appropriate for HC dialogue. That said, we lack an accepted computational model of grounding and turn-taking in HC dialogue; or more precisely, we lack an accepted (or to the author's knowledge, proposed) computational model of grounding and turn-taking in the presence of the asymmetric ASR channel, one of the focuses of this work.

Chapter 3: Current approaches

In reviewing techniques, it is useful to distinguish the *type* of uncertainty a particular approach is addressing. We use the following taxonomy:

1. *User-belief uncertainty*: Uncertainty in the user’s beliefs
2. *ASR-channel uncertainty*: Uncertainty in what the user said
3. *Dialogue-state uncertainty*: Uncertainty in the state of the conversation
4. *Action-affect uncertainty*: Uncertainty in the affect of the system’s actions

3.1 Traditional methods

SDSs have long had access to basic *confidence measures*, usually derived from the log-likelihood score of the speech recognition result. A confidence measure attempts to quantify the certainty in the recognition result. Dialogue managers usually specify a *confidence threshold*, and utterances with confidence scores below this threshold are *rejected* – i.e., the system discards the recognition result, often responding with something like “Sorry, I didn’t understand you.”

When a recognition result scores above a confidence threshold, there is still a non-trivial possibility that it is incorrect. Thus SDS designers use *confirmations* to reduce ASR-channel uncertainty. Because confirmations are perceived to increase the duration of dialogue and decrease user satisfaction,⁹ a distinction is sometimes made between *implicit confirmations* (which repeat the system’s guess of the user’s utterance/intention and move on without requiring a user response) and *explicit confirmations* (which ask the user a direct question such as “Did you say *London*?”). Designers may select one of these confirmation types at design time based on expected accuracy rates, or some notion of the “cost” of a recognition error; alternatively, the system may select one at run-time using the confidence score, selecting implicit confirmation in the presence of higher confidence scores and explicit confirmations in the presence of lower confidence scores.

Designers employing traditional methods often deliberate on how much information to gather before confirming – confirming more items is faster in the presence of good recognition, but more time-consuming to correct in the presence of poor recognition.

⁹ As discussed in Section 0, we shall see these intuitions are not borne out by user satisfaction studies.

Correction strategies may make use of *skiplists*, which exclude previously dis-confirmed items from the *N-Best list* or *word-lattice* in subsequent recognitions.

The simplicity of these methods combined with careful system prompt wording and delivery have enabled the deployment of large, successful commercial systems. In addition, because traditional methods are relatively well understood, they can be easily incorporated in domain-independent ways into a variety of rule-based dialogue management frameworks. For examples, see the Mercury flight reservation system (Seneff and Polifroni, 2000), the expressive and flexible TRINDI framework (Larsson and Traum, 2000), the *Activity Model* approach (Lemon et al., 2002b), or rapid development frameworks such as RAVENCLAW (Bohus and Rudnicky, 2003) and Variant Transduction (Alshawi and Douglas, 2001), and others such as (Denecke, 2002).¹⁰

3.2 Better confidence measures

Several researchers have sought to derive better confidence measures by incorporating additional information. These methods all attempt to find a better characterisation of user-action uncertainty in an effort to reduced uncertainty in the recognition result.

(Carpenter et al., 1999) uses a Bayesian Belief Network to produce a more robust confidence score. They find that a network taking slot bigram score, acoustic confidence, a parse coverage metric, the current state, and its expected slot outperforms the acoustic confidence alone.

(Krahmer et al., 1999) and (Krahmer et al., 2001) apply supervised learning to predict misrecognitions. Their feature set is inspired by human/human patterns of error correction and includes cues such as shorter vs. longer turns, signs of affirmation vs. disconfirmation, presence of an answer to a subsequent question vs. lack thereof, changes to a previously provided answer (i.e., corrections), repetitions, and presence of new information.

(Hirschberg et al. 2001) and (Litman et al., 2001) also use supervised learning to identify “aware sights”, defined as “turns where a user, while interacting with a machine, first becomes aware that the system has misrecognised a previous user turn.” They use prosodic and other run-time features as inputs to derive a RIPPER-based classifier.

(Litman and Pan, 2000) create a cumulative measure of recognition performance. They apply supervised learning on a corpus to predict “good” (<11% error turns) vs. “bad” dialogues (>11% error turns) using features such as: acoustic confidence, measures of dialogue efficiency, naturalness (e.g., number of user requests for help), experimental parameters (e.g., initial dialogue strategy configuration), and lexical clues (e.g., lexical items in ASR output).¹¹ A dialogue manager is implemented which changes strategy

¹⁰ Many of these frameworks could take advantage of statistical techniques. For example, (Larsson and Traum, 2000) and (Bohus and Rudnicky, 2003) both indicate that rule selection could operate probabilistically in their frameworks. The point is that example systems built with the frameworks all use traditional techniques.

¹¹ Interestingly, the best ruleset includes only a cumulative measure of acoustic confidence, indicating that it is by far the single most informative feature.

based on this cumulative confidence measure, and the resulting system is shown to improve task success as compared to a non-adaptive version of the same system.

(Langkilde et al., 1999) demonstrate that a problematic dialogues can be predicted with 70-80% accuracy in the first 1-2 exchanges using supervised learning (RIPPER). Features found to best predict problematic dialogues include: ASR duration, ASR confidence scores, requests for help, and a measure of the grammar coverage.

3.3 Localised goal and dialogue act detection

We may posit the presence of a hidden user goal or dialogue act from which an utterance arises.¹² When we limit ourselves to examining a small number of user utterances in one vicinity to determine goal or dialogue act, we call this *localised goal/dialogue act detection* (as compared to modelling an evolving goal over the course of a dialogue, explored below).

Localised dialogue act detection (also called dialogue act classification) has been extensively studied in both HH and HC dialogue. Numerous dialogue act taxonomies exist -- see (Stolcke et al., 2000) for an example list of taxonomies, and (Traum, 2000) for relevant issues in developing taxonomies. The reliability of a dialogue act taxonomy (or a particular application of one to a corpus) is typically assessed with the Kappa statistic (Carletta, 1996). Trained annotators rarely show high agreement ratings, pointing to a basic trade-off between the expressive power of a taxonomy (number and granularity of the acts) and its reliability.

Even so, dialogue acts provide a useful, generative model of an infinite set of utterances, particularly relevant in the (HH) machine translation domain. (Stolcke et al., 2000) finds that an HMM model can predict dialogue acts nearly as well as human annotators, though also finds that inclusion of dialogue act information into the speech recognition decoding process doesn't improve accuracy significantly.

In the HC domain, (He and Young, 2003) infers dialogue acts from the *concepts* expressed in a single utterance by learning a Tree-Augmented Naïve Bayes (TAN) algorithm. The authors apply the system to the ATIS domain with positive results.

Localised goal detection has been studied in detail in the call-routing domain. (Gorin et al., 1997) extracts salient phrase fragments to infer goal; (Chu-Carroll and Carpenter, 1999) uses a vector-based approach. (Ng and Lua, 2003) considers multiple hypotheses from the speech recogniser and applying them to a Transferable Belief Models.

3.4 Ranked state hypotheses

(Lewin, 1998), working on the Autoroute task, posits a dialogue model based on rules describing conversational game theory. Assuming rational behaviour on the part of the caller and system, rules can be applied to utterances to infer beliefs. An ordering of the rules provides a preference for their application; if an utterance arrives which cannot be explained within the hypothesised user beliefs, the system may *backtrack* (using Prolog's back-tracking search) to consider different interpretations of (i.e., apply different rules to) past utterances to arrive at a new hypothesis of user beliefs consistent with the user's

¹² The division between a *goal* and *dialogue act* is necessarily grey and some overlap is inevitable.

utterance. Thus the system is effectively maintaining an ordered (but unscored) list of possible dialogue states, modelling both user-belief and dialogue-state uncertainty.

(Higashinaka et al., 2003) augments the discourse understanding (frame-based) component of a SDS to maintain an N-best list of dialogue states in a form-filling domain. Multiple state hypotheses are generated at each turn by considering multiple recognition hypotheses and multiple interpretations for each recognition hypothesis. Possible next dialogue states are generated then scored using a linear combination of previous score, acoustic score, dialogue act n-gram probability, and dialogue act likelihood given the current dialogue state. The system maintains a *dialogue beam width* of approximately 30 states. The dialogue manager is a fixed, rule-based policy which considers only the state with the highest score.

3.5 Decision-theoretic methods

Decision-theoretic methods, which employ a combination of Bayes Nets for modelling state and immediate utility to guide action, have been successfully applied to SDSs, addressing user-belief, ASR-channel, and dialogue-state uncertainty.

(Paek and Horvitz, 2000a) uses Bayesian networks to estimate the probability that the system understands the user's goal in a multi-modal, automated building receptionist. There are three groups of inputs; examples for each include:

- *Signal level*: ASR confidence, threshold setting, overall parse fit, number of phrasal heads, number of non-terminals.
- *Maintenance level*: whether the telephone is in use, whether other people are present in the room, whether the user's utterance includes the name of system.
- *Multi-modal*: whether the user is looking at the system (*eye gaze*), whether the user is moving toward or away from the system.

Using these inputs, the dialogue model estimates:

- *Maintenance status*, which takes four values: *No Channel*, *Channel but no signal*, *Signal but no channel*, and *Channel and signal*. *Channel* refers to whether the user is directing their communication at the system; *signal* refers to the quality/confidence/presence of information which is coming from the user.
- *Intention status*, a binary attribute which models the "task of understanding the semantic content of signals."
- *Conversation status*, a binary attribute which indicates whether the conversation as a whole is on-track.

Two nodes then summarise the current state of the dialogue:

- *Grounding status*, which takes values *Ok*, *Channel failure*, *Signal failure*, *Intention failure*, and *Conversation failure*.

- Activity goal, which indicates whether the user is participating in a joint activity with the system or with someone else

Utility is computed as a function of grounding status, activity goal, and *grounding strategy*, which the dialogue manager selects to maximise utility.¹³ After selecting a grounding strategy, the system selects a specific focus for the repair using a Value-of-Information (VOI) metric (e.g., “Are you talking to me?” vs. “Could you repeat that?”).

(Horvitz and Paek, 2000) applies this model to the *DeepListener* project, which offers assistance to users while interacting with a desktop software application.

The dialogue model uses a two-stage temporal Bayes Net to form a probability distribution over a user’s goal. Input includes various GUI observations such as menu selections in addition to speech observations. The dialogue model forms a distribution over *user’s spoken intention*, such as *Acknowledgement* (accept offered help), *Negation* (decline offered help), *Reflection* (deliberating assistance for help), *Unrecognised signal*, and *No signal*. Utility is informed by this distribution over user’s goal and the system action – possible actions include *Ignore*, *Invoke offered help*, or *engage in a clarification dialogue*.

(Paek et al., 2000) describes a system which acts as a presentation assistant, listening for commands to advance the slide during a business presentation. Observations again include a variety of ASR features and eye gaze. This system focuses on the *Signal but no channel* maintenance level, differentiating when speech is intended for the computer.

Modelling distributions over dialogue state and user goal allow informed, principled decisions to be made about dialogue-state uncertainty, user-belief uncertainty, and ASR-channel uncertainty. Segmenting the joint probability into many conditional probabilities using a Bayesian Network reduces the number of required parameters, and also allows designers to use intuitions about dependencies between the state of the conversation, user’s actions, and observations.

3.6 Markov Decision Processes (MDPs)

Introduction to Markov Decision Processes

Markov Decision Processes (MDPs) provide a model for intelligent behaviour in a temporal process. See (Sutton and Barto, 1998) for a good introduction. An MDP is defined by a tuple $\{S, A, T, R\}$ where S is a set of states, A is a set of possible actions an agent may take, T defines a transition probability $p(s'|s, a)$ (sometimes called the transition matrix), and R defines the expected (immediate, real number) reward $r(s, a)$.¹⁴

At each time-step, the agent is in a particular state s . The agent selects an action a , receives a reward r , and transitions to a state s' , where s' depends on the current state and the action selected. The goal of the agent is to maximise the cumulative reward over time (sometimes called the return), subject to a geometric discount factor λ , $0 \leq \lambda \leq 1$. A

¹³ The utilities themselves “can be elicited from users through psychological experiments or direct assessment tools.”

¹⁴ The reward function may also be a function of the next state s' .

policy, $\pi(s)$, is a mapping from states to actions; an *optimal policy*, $\pi^*(s)$, is one which maximises cumulative reward over time.

MDPs are intuitively appealing for dialogue management as they model system-action uncertainty in a principled way. In addition, this approach allows designers to specify rewards for desired and undesired outcomes (e.g., a caller hanging up, successfully completing a task) without specifying how to achieve goals. Further, there are a variety of well-understood solution techniques, including approximations for large state spaces and strategies for learning the transition matrix in parallel with an optimal policy.

Overview of SDS/MDP literature

(Levin and Pieraccini, 1997) is the first presentation of dialogue management framed as an MDP. (Levin et al., 1998) and (Levin et al., 2000) provide a formal treatment of how an MDP may be applied to the dialogue management task. The authors test the formulation with a simulated ATIS task using a simple user model. The objective is to find optimal tradeoffs among (1) the costs of querying the user to restrict (or broaden) their flight search, and (2) the costs of presenting too many (or too few) flight options, and (3) the costs of accessing a database.¹⁵

(Walker et al., 1998b) constructs a SDS with an MDP-based dialogue manager, with the objective of finding optimal tradeoffs between choice of initiative (system or mixed), and choice of information presentation style for reading and summarising messages. A reward measure derived from the PARADISE method (see section 4.3) is used. Interestingly, the system initiative action was found to maximise the reward measure.

(Singh et al., 2002) similarly constructs a real system with an MDP dialogue manager. The authors' objective is to find a confirmation and initiative strategy which balances number of turns with accurately obtaining several pieces of information. 311 dialogues (54 participants, 6 tasks) were collected with a random policy to perform training. The resulting optimal policy was evaluated by collecting 124 dialogues (different 21 participants, same 6 tasks). Because each user interacted with the system multiple times, there was a strong user-learning effect. For experienced users, the optimal policy led to more successful dialogues than the random policy; surprising, the random policy had a higher task completion rate in the 1st and 2nd tasks. Subjective evaluations were not significantly different between the optimal and random policies.

¹⁵ The authors note: "It is impossible to use supervised learning with a given corpus of interactions to estimate the parameters or the optimal strategy of a dialogue system. In fact, supervised learning techniques are based on the assumption that both the training and the test examples are drawn independently from the same distribution. In the case of MDP this assumption is incorrect if the training data is composed of complete sessions produced using a fixed strategy, like in the case of dialogue, since the strategy itself determines the distribution of states in the corpus. Therefore, even though we could in principle use supervised learning to estimate the strategy in the corpus very accurately, a small deviation in the learned strategy might produce a distribution over the state space completely different from the one observed in the training set, for which we do not have accurate prediction of strategy. In reinforcement learning, the optimal strategy (or the model parameters) is learned not from a static corpus but through interaction."

Training issues

Collecting sufficient training episodes is an acute problem for MDPs. Back-off schemes and user models have been proposed.

(Goddeau and Pineau, 2000) acknowledges the practical limitations of training an MDP, and proposes assigning each state a “back-off” state for the purposes of learning transition probabilities. This allows an optimal policy to be estimated sooner, and refined as data sparsity is reduced over time.

(Pietquin and Renals, 2002) creates a user model for a computer-buying dialogue, similar in structure to the ATIS task. The user model includes a rather detailed concept-level confidence score and occasionally provides too much information, simulating a mixed-initiative-like behaviour. The optimal policy asks for values which are more likely to be recognised first.

(Scheffler and Young, 2002) creates an advanced, detailed model of user behaviour at the dialogue act level, including a model of ASR errors. The authors then apply this model to a variety of state space representations and cost functions to find optimal policies. Automatically generated policies often out-perform hand-crafted designs. The authors use comparatively sophisticated learning techniques (*eligibility traces*) which attempt to partially overcome the fact that the state is not fully observable.

State space and action set selection

Most systems’ state spaces are based on vectors describing *information slots*. A variety of action sets have been explored.

(Levin and Pieraccini, 1997) formulate two different state space models. In a toy example, states are vectors consisting of slots, where each slot may be filled or unfilled. In an ATIS example, states are vectors consisting of the following fields: the current request, the current database query, the number of tuples in the DB query, possible additional constraints and possible constraints to relax. Actions include selecting parameters to narrow or relax the search criteria to decrease or increase the number of tuples in the current DB query.

(Walker et al., 1998b) does not fully describe the state space used. The action set consists of (1) selecting between a system initiative and mixed initiative strategy, (2) choosing 1 of 3 strategies for reading messages, and (3) choosing 1 of 3 strategies for summarising a list of messages when the caller says “summarise my messages.”

(Singh et al., 2002) creates a handcrafted state-space of 42 states derived from reasonable permutations of the following vector:

- *Greet*: Has the user been greeted yet? (y/n)
- *Attribute*: Which slot is currently active (0,1,2,3,4)
- *Confidence/confirmed*: 0,1,2 for low, medium, high ASR confidence; 3,4 for explicitly confirmed and disconfirmed.
- *Value*: Whether a value has been obtained for the current attribute (y/n)

- *Tries*: How many times current attribute has been asked (0,1,2)
- *Grammar*: Whether non-restrictive or restrictive grammar was used (0,1)
- *History*: Whether there was any trouble on any previous attribute (0,1)

Actions include using a System, Mixed, or User initiative prompt to ask (or re-ask, or confirm) the first two data fields. To limit data sparsity, actions were limited to 2 choices in any given state.

The state space described in (Pietquin and Renals, 2002) is a vector of slots where each is associated with a confidence – empty, low, or high. The action set includes greeting, ask, confirm, relax, db query, and close.

(Scheffler and Young, 2002) examine 5 different state space alternatives. Each includes the current slot in focus, the status of that slot, the number of empty slots, and the number of unconfirmed slots. State space variants optionally include parameters describing how the current slot was obtained and various confidence measures. Actions include 2 different types of questions, 3 different types of confirmations, and accepting the current slot.

While a variety of state space configurations have been explored, researchers have struggled to cope with the “curse of dimensionality” in constructing state spaces. Researchers have responded to date by hand-crafting state spaces which limit size and action choice to maintain tractability. For larger problems this approach becomes unworkable. One principled solution to coping with larger state spaces is the use of a parameterised value function (Sutton and Barto, 1998), which does not appear to have been applied to the literature yet (proposed, but not addressed, in (Williams and Young, 2003)).

In addition, one oft-cited goal of MDP-based dialogue managers is reducing human designer time. However, by hand-crafting a state space and action set, designers risk increasing rather than decreasing their design workload.

Reward functions

(Levin and Pieraccini, 1997) suggest two different reward functions. In a toy example, the authors use: number of turns, probability of a misrecognition associated with a given interaction, and a penalty for submitting an incomplete form. In an ATIS example, the authors use: number of turns, cost of data access, and penalties for returning no information or too much information to the user.

(Levin et al., 1998) bases reward on a weighted sum of (1) number of user interactions, (2) expected cost of data retrieval, (3) cost of data presentation, and (4) "expected distance from the application goal" – a penalty for not providing any information to the user.

(Walker et al., 1998b) applies a dialogue-final measure of user satisfaction derived from PARADISE evaluations – i.e., $\text{Reward (at the end of the dialogue)} = 0.47 * (\text{Mean recognition score}) + 0.21 * (\text{Perception of task completion}) - 0.15 * (\text{elapsed time})$

(Singh et al., 2002) selects a reward based solely on correct or incorrect submission of the user’s information.

(Pietquin and Renals, 2002) use a reward function that is a weighted sum of per-turn penalties, number of database accesses, number of presented records, confidence level of the current user’s utterance, and a “function of the modelled user’s satisfaction.”

(Scheffler and Young, 2002) uses a per-turn penalty and a task failure penalty, assessed at the end of the dialogue, citing simplicity. While acknowledging that an ideal cost function should be based on user satisfaction, the authors suggest a two stage process, first creating a system based on objective metrics, then deploying the system to users to collect preference/satisfaction data.

There is no accepted formulation for a reward function. While per-turn penalties (or discounting) are necessary to induce systems to complete tasks, studies have shown that user satisfaction does not correlate with task completion times (see section 4.3). Usage of features derived from user satisfaction studies seem most appropriate, but are harder to apply in practice as they require either more annotation or collecting user satisfaction scores. Lastly, it is an open question how to combine system-related costs (e.g., database access penalties) and user-centred rewards.

3.7 Partially Observable Markov Decision Processes (POMDPs)

Introduction to Partially Observable Markov Decision Processes

Partially-Observable Markov Decision Processes (POMDPs) extend Markov Decision Processes by removing the requirement that the system knows its current state precisely. Instead, the system makes *observations* about the outside world which give incomplete information about the true current state. The system maintains a *belief state* – a distribution over MDP states – in the absence of knowing its state exactly. See (Kaelbling et al., 1998) for a good introduction.

Formally, a POMDP extends an MDP by adding a (finite) set of observations O , which occur according to a distribution $p(o | a, s')$, sometimes called the *observation function*.¹⁶ We replace s – the true current state, which is now unobservable, with $b(s)$ – a distribution over all states.

The immediate reward is computed as the reward function on belief states, $\rho(b, a)$:

$$\rho(b, a) = \sum_{s \in S} b(s)r(s, a)$$

At a new timestep, the new (exact) belief state $b'(s')$ can be computed as:

$$\begin{aligned} b'(s') &= p(s' | o, a, b) \\ &= \frac{p(o | s', a, b)p(s' | a, b)}{p(o | a, b)} \end{aligned}$$

¹⁶ This can be read as “The probability of making an observation given that action a was taken, causing a transition to state s' .”

$$\begin{aligned}
& p(o | s', a) \sum_{s \in S} p(s' | a, b, s) P(s | a, b) \\
= & \frac{p(o | s', a) \sum_{s \in S} p(s' | a, b, s) P(s | a, b)}{p(o | a, b)} \\
& p(o | s', a) \sum_{s \in S} p(s' | a, s) b(s) \\
= & \frac{p(o | s', a) \sum_{s \in S} p(s' | a, s) b(s)}{p(o | a, b)}
\end{aligned}$$

The numerator is the observation function, transition matrix, and current belief state. The denominator is independent of s' , and can be regarded as a normalising factor.

A *policy* in a POMDP is a partitioning of belief space, $\pi(b)$, which maps regions of belief space onto actions. Finding exact optimal policies is intractable for all but the simplest problems – generally those problems with less than 10 actions, states, and observations. As a result, most solution methods are approximate. See section 4.3 for a short discussion.

Augmented MDPs

(Roy et al., 2000) applies an *Augmented MDP* to the dialogue manager of a robot in a nursing home environment. The underlying MDP consists of 13 states and 20 actions, 10 of which are performance-related (e.g., go to a different room, output information) and 10 of which ask clarifying questions. The dialogue model includes 16 observations – 15 keywords and 1 non-sense word. The reward function returns -1 for each turn, +100 for the correct fulfilment of the user’s request, and -100 for incorrect fulfilment.

The authors note that a complete solution to POMDPs for a state space of 7 states is possible, yet an exact solution is already intractable for 13 states. As a result, the authors elect to use an *Augmented MDP*, which approximates belief space as (1) the state with the single highest probability and (2) the entropy of the state space. A policy in an Augmented MDP is thus a partitioning of only one real-numbered dimension (i.e., entropy) for each state.

The reward gained per dialogue was significantly better for the Augmented MDP than for the MDP trained on the same data. The authors note that the Augmented MDP uses confirmation much more aggressively, so average turns/dialogue is higher for the POMDP than for the MDP; however, the variance for POMDPs is lower.

POMDPs with approximate solutions

(Zhang et al., 2001) creates a POMDP to model a tour guide SDS. The underlying MDP consists of 30 states, comprised of two vector components: (1) the user’s goal (or request), which takes one of 6 possible values: *s-visit-gate*, *s-visit-hall*, *s-ask-gate-height*, *s-ask-gate-size*, *s-ask-hall-height*, and *s-ask-hall-size*; and (2) a “hidden” state – effectively the current dialogue state – taking one of 5 values: *s-normal*, *s-silent*, *s-error-noisy*, *s-error-silent*, and *s-error-overheard*.¹⁷ The product of these two vectors gives 30 states.

The system has 18 actions:

¹⁷ Labels indicated here have been created for this exposition – labels are not given in the paper.

- 6 domain actions are associated with satisfying the user's goal: *a-visit-gate*, *a-visit-hall*, *a-ask-gate-height*, *a-ask-gate-size*, *a-ask-hall-height*, and *a-ask-hall-size*
- 12 dialogue repair actions:
 - Ask the user to repeat themselves: *a-repeat*
 - Ask the user's intention: *a-ask-or-visit*, *a-height-or-size*, and *a-gate-or-hall*
 - Declaring the user's intention: *a-declare-visit-gate*, *a-declare-visit-hall*, *a-declare-ask-gate-height*, *a-declare-ask-gate-size*, *a-declare-ask-hall-height*, and *a-declare-ask-hall-size*
 - *a-ignore* (i.e., do nothing)
 - *a-troubleshoot*

The transition function assumes that repair actions do not change the user's goal, and that the hidden system state changes only with repair actions.

There are 11 rewards defined:

- 7 rewards associated with repair actions:
 - An immediate reward for taking the action *a-repeat*
 - An immediate reward for asking the user's intention (e.g., *a-ask-or-visit*)
 - An immediate reward for declaring the user's intention (e.g., *a-declare-visit-gate*)
 - An immediate reward for taking *a-ignore* when in state [**,s-silent*], and another for taking *a-ignore* when not in that state.
 - An immediately reward for taking *a-troubleshoot* when in state [**,s-error-**], and another for taking *a-troubleshoot* when not in that state.
- 4 rewards associated with domain actions:
 - A reward for taking the wrong domain action
 - A reward for taking the (completely) correct domain action
 - Two different rewards for taking domain actions which are partially correct

Finally, there are 25 observations, which are the output of a Bayesian network:

- *no-signal-and-no-channel*, *no-signal*, and *no-channel*
- *yes* and *no*
- 6 complete requests
- Incomplete requests (a total of 14 permutations)

Many of the probability functions are hand-crafted. A discount factor of 0.9 is used and a fixed, uniform initial distribution over user goals is assumed.

When the system runs in real-time, there is another Bayesian network that is used to determine when the user has barged-in; however assessment in the paper is performed with a user model.

Exact algorithms are unable to find a solution. The authors explore the following approximations and assess them with respect to running time, value function quality, and policy quality: the MDP approximation, the Q-MDP approximation, the Fast Informed-Bound (FIB) approximation, and grid-based approximation. MDP, Q-MDP, and FIB all assume that the uncertainty in the future will “disappear”; thus, they rarely/never take information-gathering actions. Grid-based approximation (which computes an approximate value function in quantised/partitioned belief space) is found to yield the best rewards over time, but also consumes the most computational resources.

3.8 Summary of current approaches

Table 1 summarises current approaches to modelling uncertainty in spoken dialogue systems.

Of all the approaches, only POMDPs model all uncertainties explicitly.

Note that MDPs and POMDPs are the only two approaches to explicitly model the uncertainty in their actions – this is tantamount to planning under uncertainty. In order to look into the future, their state spaces must encode both a dialogue model as well as a *user model*.

As noted above, user models have been studied somewhat extensively in relation to MDPs – (Scheffler and Young, 2001) and (Eckert et al., 1998) develop sophisticated *external* user models. One might observe that a (PO)MDP will “learn” a transition function that approximates the user model, and may ask whether it would be more straightforward to embed a model of user behaviour into the transition function directly. For an MDP, direct encoding isn’t possible, primarily because MDPs cannot explicitly model the user’s state, since it is unobserved. However, POMDPs do not suffer this limitation, and probabilistic user model information may be incorporated directly.

Further, POMDPs and Decision-Theoretic methods differ only in their ability to reason about the future. Because Decision-Theoretic-based dialogue managers use immediate utility and don’t include a model of user behaviour/reaction, they can’t be said to be reasoning about the future.

Approach	Type of uncertainty			
	Action-affect	User-belief	ASR-channel	Dialogue-state
Traditional methods	–	–	Binary or handcrafted classes segmented using handcrafted thresholds	–
More robust confidence measures	–	–	Binary or handcrafted classes using SL-derived thresholds	–
Localised goal/dialogue act detection	–	Localised (single-utterance) distribution	Localised distribution; possibly considering multiple hypotheses	–
Ranked state hypotheses	–	–	Multiple (ordered, scored) hypotheses; multiple abstract intentions	Ordered (scored) list
Decision-theoretic methods	–	Explicit distribution	Explicit distribution	Explicit distribution
MDPs	Maximum cumulative reward	–	(Same as traditional methods)	–
POMDPs	Maximum cumulative reward	Explicit distribution	Explicit distribution ¹⁸	Explicit distribution

Table 1: Comparison of current approaches to modelling uncertainty in SDS

¹⁸ Limited to a finite set of observations

Chapter 4: Project proposal

Section 2 explored the observed differences between HH and HC dialogue, and concluded that the patterns of grounding and turn-taking were significantly different. This work asserts that for SDSs to progress, a well-founded model of grounding and turn-taking in the presence of the ASR channel is required.

Exploring dialogue models in HC conversation is challenging because of a basic chicken-and-egg problem: To evaluate a model, we need to see how well it follows a corpus; to collect a corpus, we need a working system; and to build a working system, we need to select a dialogue manager and a model. On the one hand, the corpus must reflect all/most dialogue situations (i.e., sequences of user and system actions), including all types of misunderstandings; on the other hand, for the resulting user model to be representative of a user's behaviour with a reasonably competent system, the system must be somewhat effective (i.e., not perceived to be random). In addition, ideally we'd like to somehow "discover" an appropriate action set rather than pre-select and evaluate.

In this work we propose a different approach. We strive to understand the affects of the ASR channel *in the abstract* without assuming a dialogue model or dialogue manager at all. Instead, we propose allowing two *people* to communicate through a simulated ASR channel. To the best of the author's knowledge, this type of experiment has not been conducted (at least not published) before. We call this endeavour phase 1; section 4.1 provides details.

In phase 2, we propose annotating and examine the resulting dialogues to iteratively propose and evaluate dialogue models. Section 4.2 provides details.

Based on the interaction data collected, we can propose and evaluate probabilistic user models. Then, by combining a dialogue model *and* a user model to form a POMDP, we can construct a dialogue manager. This comprises phase 3, described in section 4.3.

As an illustration and proof-of-concept for phases 2 and 3, a Toy POMP has been created. Its components are described as they become relevant to the proposal.

4.1 Project proposal, part 1: Data collection

Experimental design

As discussed in section 2, there are three basic sources of uncertainty in the ASR channel: removal of meaningful prosodic information, word insertions/substitutions/deletions, and turn-taking errors/response delays. Further, because we want to discover what actions are

most appropriate, we allow the wizard to respond as flexibly as possible – i.e., the wizard does not select from a pre-defined list of actions.

It was contemplated whether the affects of word substitutions/deletions/insertions and the removal of meaningful prosodic information could be accomplished by some kind of acoustic transformation rather than conversion to text. Up to a certain noise threshold, the human ear has excellent ability to reconstruct speech signals, beyond that, recognition declines almost instantly. Further, it is unclear how to remove prosodic information. Conversion to text removes prosodic information entirely, and allows errors analogous to speech recognition errors to be simulated directly.

It was further contemplated whether a real speech recogniser should be used, or a human *typist*. To isolate the effects of removal of prosodic information and turn-taking effects, we'd like to explore a continuum of word error-rate (WER) conditions, from 0% WER to a high WER. In real time, WER can be increased without a typist, but not decreased. Thus to study low/zero WER conditions, we need a human typist.

The subject receiving uncorrupted input is called the *user*; the subject receiving corrupted input is called the *wizard*. A logical depiction of the experimental set-up is given in Figure 1. All aspects of the session will be logged, including video tapes of both the user and wizard.

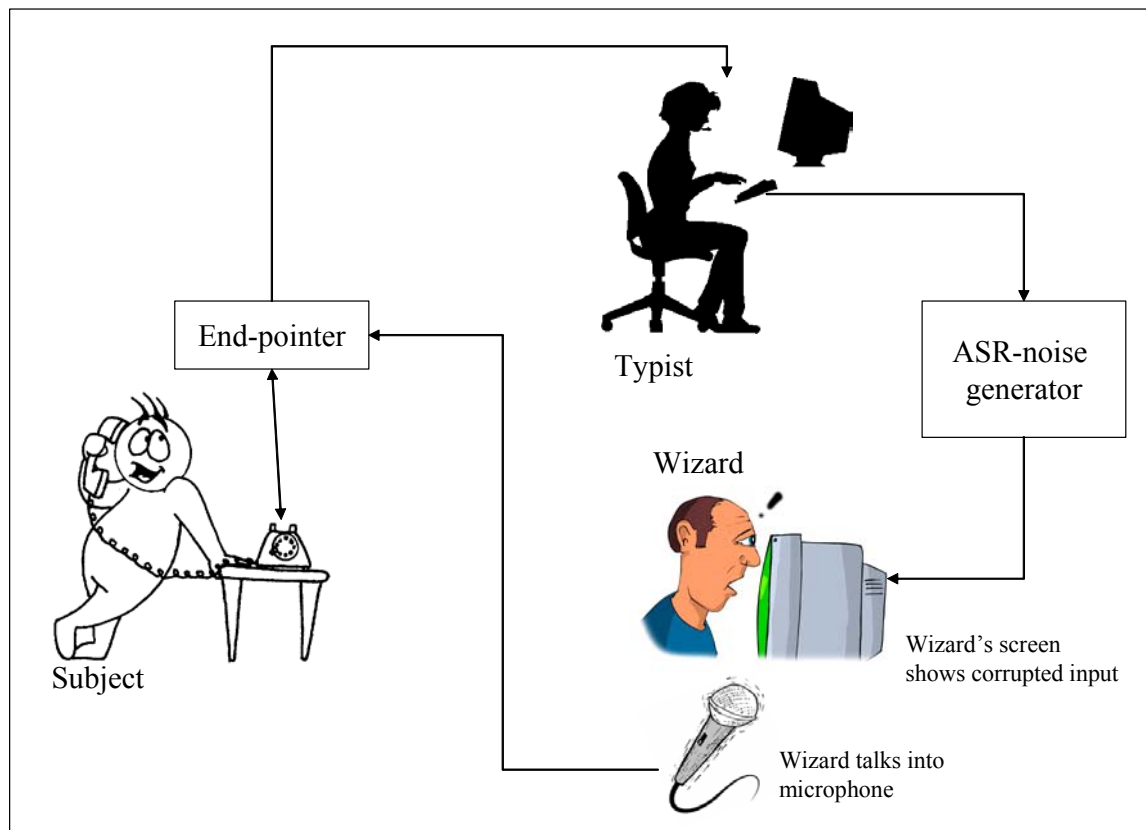


Figure 1: Experimental design for data collection

Experiments in which a typist has replaced (or augmented) the speech recognition component have been successfully completed in the past; for example, see (McInnes et al., 1999).

It is hoped that this corpus may be of use to other researchers; procedures for sharing it will be explored.

Theoretical design considerations

To simulate the turn-taking noise of the ASR channel, a real end-pointer will be used to segment user utterances and to control barge-in.

It has been contemplated whether the user should be told they're communicating with a person or a machine. Since we're trying to understand the affects of the ASR channel alone, the user will be told they are communicating with another person. However, it is important that they believe they are using a real speech recogniser – so they will be told that the purpose of the experiment is to evaluate a speech recogniser.

It was contemplated whether the wizard's speech should be rendered as TTS, as is common in SDSs. We note that typing and generating TTS would add considerable (possibly non-realistic) per-turn delay; further, since written language is different than spoken language, typing might introduce language that a wizard is unlikely to produce in spontaneous dialogue. The wizard will speak into a microphone to the user.

A total of 48 different tasks will be created and used for each wizard so that wizards do not learn what task to expect. Subjects (wizards and users) will be asked Likert and free-response questions about their experiences after each dialogue rating their satisfaction with the experience, whether they were successful, etc.

The word substitution/deletion/insertion probabilities will be estimated from the output of a real recogniser.

Practical design considerations

A total of 64 participants (not counting the typist) will be required; see Table 2.

WER	Wizards	Users/wizard	Dialogues/User	Users/WER	Dialogues/WER
0%	4	4	4	16	64
5%	4	4	4	16	64
15%	4	4	4	16	64
30%	4	4	4	16	64
TOTALS	16			48	256

Table 2: Data collection experiments

It is estimated that each dialogue will take approximately 15 minutes. Thus one user session will take approximately 1 hour, and one wizard session will take approximately 4 hours (likely spread out over a morning, an afternoon, or whole day).

The task is to be determined. To allow comparison with existing work, a domain like airline reservations would be desirable; however, this domain would require some wizard

training, and possibly a computer system for the wizard to use (though a commercial website could be used).

Subjects, wizards, and typists will need to be compensated. An example budget is given in Table 3.

Role	Number	Hours/person	Cost/hour	Total
User	48	1	GBP 20	GBP 960
Wizard	16	4	GBP 20	GBP 960
Typist	1	48	GBP 20	GBP 960
TOTAL				GBP 2880

Table 3: Data collection costs

It is envisaged that subjects and wizards will be recruited from the undergraduate and graduate student population of the University of Cambridge. Typists will be hired either free-lance or through a recruiting agency.

The author will create the simulation system using equipment available in the Cambridge University Engineering Department. Three small nearby rooms will be required.

4.2 Project proposal, part 2: Dialogue model creation

Data preparation

First the spoken portion of interaction data will be transcribed to plain text. Next, one of the wizard dialogues from each subject will be held out as test data and not consulted again until the testing phase; the remaining data will be segmented into a training set and a validation set.

A variety of salient features for each user utterance will be enumerated, such as user utterance length, number of words in user utterance recognition result, whether barge-in was triggered, number of words in system utterance, and acoustic score.¹⁹ In addition, a parser will be created and various features (e.g., % coverage) will be extracted.

Finally, the corpus will be annotated for dialogue acts. There are many existing dialogue act annotation schemes available – (Larsson, 1998) and (Stolcke et al., 2000) give summaries and comparisons. A goal of this work is to discover a domain independent set of grounding and turn-taking acts in the ASR channel, so different act taxonomies will be explored.

There are a variety of annotations tools freely available – for example (Orasan, 2003) and (Muller and Strube, 2003). It will be preferable in this case to annotate dialog acts with two annotators and calculate inter-annotator agreement (i.e., Kappa score), but it is not vital since a poor annotation will not over-predict the performance of the dialogue model.

The author has professional experience with orthographic utterance transcription, and has this year gained experience annotating the Autoroute corpus for dialogue acts (Williams and Young, 2003).

¹⁹ The ASR error simulation will also estimate acoustic score

Data preparation: Toy example

The Toy POMDP was created to feasibility test the ideas in this proposal. The Toy POMDP models a travel dialogue in which the caller wants to travel between two cities; in this world there are three cities, named City A, City B, and City C.

The Toy POMDP assumes the following were observed in a corpus:

- The actual user action, a_u . Actions are specified at the intention level. Examples include:
 - a – The user expressed “City a”
 - $to-b$ – The user expressed “To city B”
 - $from-a$ – The user expressed “From city A”
 - $from-a-to-b$ – The user expressed “From city A to city B”
 - yes, no – The user expressed yes or no
 - $null$ – The user didn’t respond
- The observation, o . Observations include all user actions, appended with “- h ” or “- l ” to indicate a high or low confidence score. For example, the observation $from-a-to-b-h$ is the observation that the caller’s intention is to go from City A to City B with a high recognition confidence score.
- The *corruption model*, given as:

$$p(o | a_u)$$

The corruption model gives the probability of making an observation given a user action. In the Toy POMDP, all substitutions are equally likely, but a substitution in a low confidence observations is more likely than a substitution in a high-confidence observation.

- The system actions, a_s . Examples include:
 - $greet$ – greet the caller and ask “How can I help you?”
 - $ask-from/ask-to$ – ask where the caller wants to go from/to
 - $conf-from-a/conf-to-b$ – confirm that the caller wants to go from City A or to City B
 - $submit-a-b$ – place an order for a ticket from City A to City B
 - $fail$ – give up the current dialogue

The dialogue proceeds as the sequence:

$$\{\dots, a_u, a_s, a'_u, a'_s, \dots\}$$

Iterative dialogue model development

Next, a state space describing the grounding & turn-taking process will be proposed. If necessary, the dialogues will be annotated with current dialogue state if dialogue state cannot be gleaned deterministically from the dialogue act annotation. A Bayesian network will be proposed which accounts for the dialogue act and current state of the dialogue. The network will be trained using the training set of annotated dialogues, and tested using the validation set. It is envisaged that multiple grounding-act and state-space models will be trialled, so several passes annotating dialogue acts and state space will be needed.

Recall that at this point, we are evaluating the performance of only a *dialogue model*, not a *dialogue manager*. Thus training the Bayesian Network with supervised learning is appropriate: the system is making a prediction about the state of the dialogue but not affecting the dialogue state.

Dialogue model development: Toy example

The Toy POMDP defines the following, simplistic dialogue state:

$$s_d \in (x_1, x_2), x_i \in \{n, u, g\}$$

Where x_1 gives the status of the departure city slot, and x_2 gives the status of the arrival city slot. n means “not said”, u means “said but ungrounded”, g means “grounded”.

Transitions between dialogue states are illustrated in Figure 2.

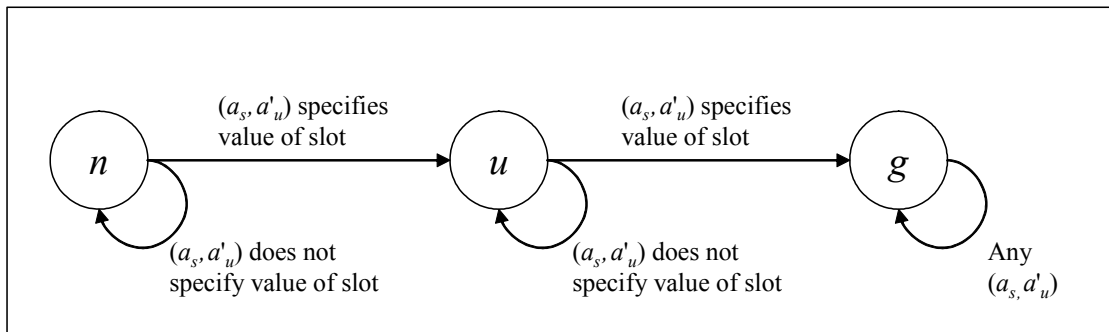


Figure 2: Finite state automaton describing grounding model for one slot (either departure city or arrival city) in the Toy POMDP. n means "not said"; u means "said but ungrounded"; g means "grounded"

Transitions between different dialogue states are expressed in the *dialogue model*:

$$p(s'_d | s_d, a_s, a'_u)$$

The dialogue model specifies the current dialogue state given the previous conversation state and the subsequent user action and system action. In the Toy POMDP, the dialogue model is deterministic – the first time a city is said, its slot is changed from n (not said) to u (ungrounded). If a city is then confirmed, or said again, it goes from u (ungrounded) to g (grounded). Subsequent mentions of a city do not change its grounding status. For example, if the state of the conversation is (n, u) , and the system asks, “Where are you leaving from”, and the caller responds “From A to B”, the new state of the conversation

is (u, g) . If the caller responded with just “A” or “from-A”, then the new state of the conversation would be (u, u) .²⁰

A Bayesian network representation of the dialogue model is given in Figure 3. Note that shaded nodes are *observable* and un-shaded nodes are *unobservable*. For a description of Bayesian Network formulation, see (Jensen, 2001).

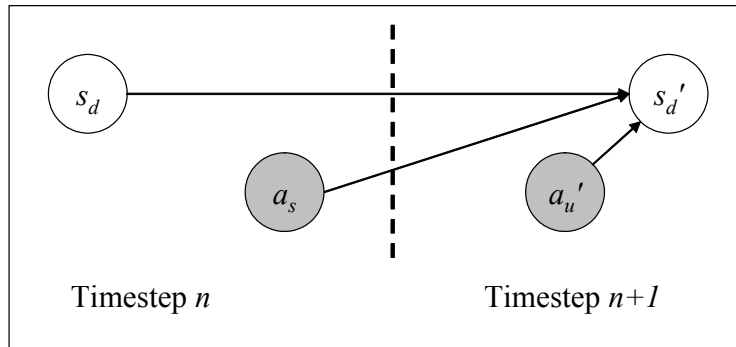


Figure 3: Bayesian network for dialogue model in the Toy POMDP (user action observable).

Model evaluation

Once a final model has been selected and evaluated on the validation data, it will then be evaluated on the held-out test data. Two evaluations will be run – one in which the user action is observable, as in Figure 3, and another in which the user action is not observable, as in Figure 4.

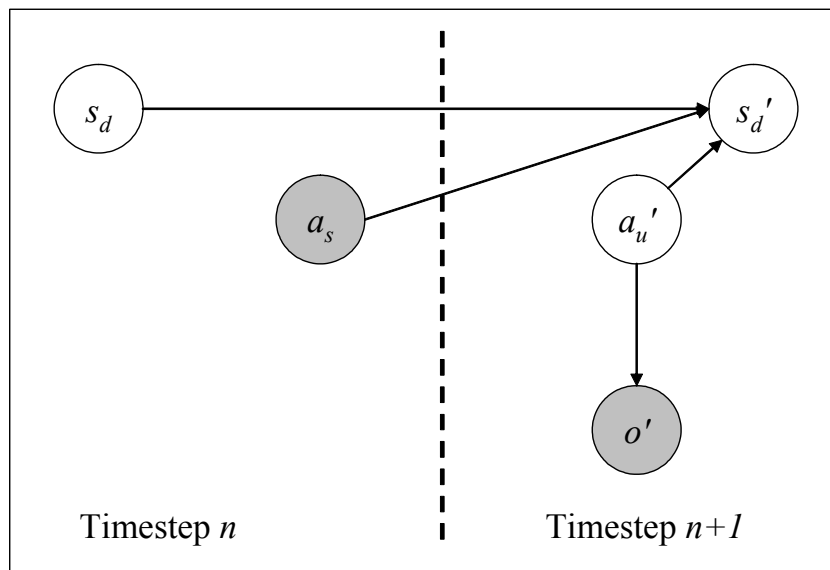


Figure 4: Bayesian Network for dialogue model in the Toy POMDP (user action not observable)

If time permits, the model could also be evaluated on other, existing HC corpora, probably requiring additional annotation. If possible, the model will be compared to any existing techniques.

²⁰ We don't need to consider the possibility of the user providing a different value since we're assuming the user belief doesn't change in the course of a dialogue in the Toy POMDP.

4.3 Project proposal, part 3: Dialogue management

Creation of user model

A dialogue manager which reasons about future states of the world needs both a dialogue model (discussed in the previous section) and a user model (discussed here).

Regardless of application, constructing a reliable user model depends on observing user behaviour in a broad array of situations – in particular for SDSs, error situations. To wit, (Eckert et al, 1997) sought to create a user model of responses to system questions based on Wizard-of-Oz experiments. The authors “realised that the corpus is not well suited to estimate these conditional probabilities. In the ATIS corpus, all information is usually given in the initial user utterance and there are rarely any follow-up utterances necessary. Thus we had to handcraft some of the probabilities but have been inspired by the characteristics of the corpus and by common sense.” This issue is addressed again in (Eckert et al., 1998). The ATIS corpus referred to is the ATIS0 Pilot corpus.²¹

Once created, the user model will be evaluated against actual user behaviour.

Creation of user model: Toy example

The Toy POMDP assumes the following *user state*:

$$s_u \in \{(a, b), (a, c), (b, a), (b, c), (c, a), (c, b)\}$$

The user state represents the user’s travel plans. For example, (a, c) indicates the user wants to travel from A to C.

The *user model* consists of two distributions. The *user belief model* is given by:

$$p(s'_u | s_u)$$

The user belief model gives the probability of a user’s belief changing over time. In the Toy POMDP model, the user’s belief is randomly assigned at the beginning of an interaction, but remains fixed throughout.

The *user behaviour model* is given by:

$$p(a'_u | s'_u, a_s)$$

The user behaviour model gives the probability of a user taking a particular action given the user’s current beliefs and the previous system’s action. In this model the user is assumed to give cooperative answers, but is sometimes inattentive, and sometimes provides too much information.²²

The components of the user model are shown in the Bayesian Network given in Figure 5.

²¹ Linguistic Data Consortium, catalogue ID LDC93S4B

²² In future work it is foreseen that the user’s action will need to be conditioned on the dialogue state as well.

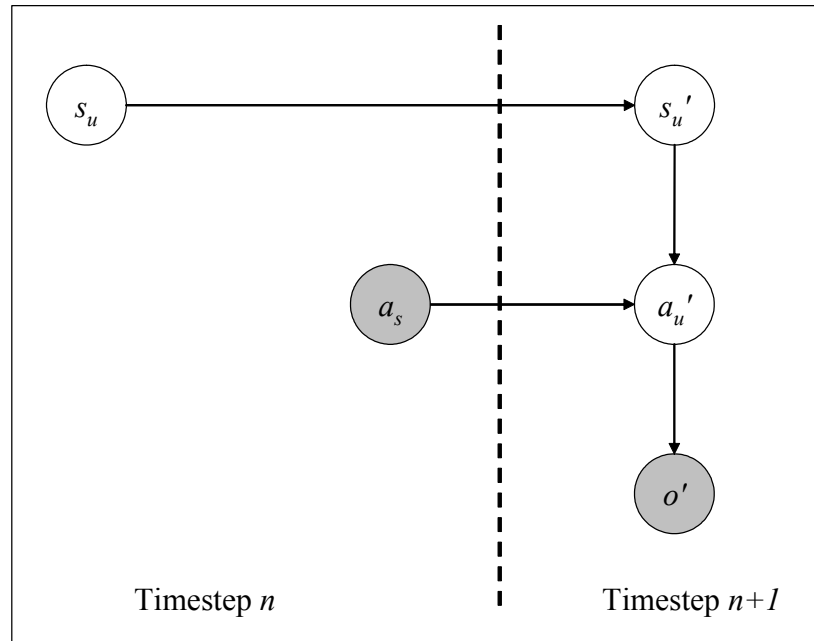


Figure 5: Bayesian network representation of the user model (user action not observable)

Specification of reward values

The final element of POMDP system design is the selection of rewards. In a POMDP, rewards can be viewed in two (non-mutually exclusive) lights:

1. Rewards give incentive for a system to take immediate actions to work toward a *long-term* goal. In this light, rewards are a catalyst for *planning*. In almost all MDP-based systems, there is a dialogue-final reward given for successful completion, high user satisfaction, etc.
2. In an MDP, actions can be limited on a per-state basis. However, in a POMDP, there is no notion of a single current (underlying) state – as such, there is no principled way to restrict action repertoire. Thus in a POMDP, rewards also indicate *preference* among immediate actions. In this sense they serve much the same capacity as utilities.

One of the strengths of the reinforcement-learning framework is the ability to directly trade-off between short- and long-term rewards. That said, one open question is whether short-term rewards will be required at all, or whether entirely appropriate short-term behaviour will result from specification of long-term objectives. Another open question is how to reward a system for “giving up” when it detects things are going horribly wrong.

Since reward measures define the optimality criteria, there is no way to directly assess whether a reward measure is “correct”. An alternative approach is to take the view that an experienced wizard’s behaviour is an example of an optimal policy, and to determine whether that policy was optimal for a given reward function. In fact, we could describe an infinitely large set of reward measures for which the wizard’s policy is optimal (this

can be derived formally). However, the calculation of a unique reward function from a given optimal policy isn't possible.

As noted in section 3.6, there are no agreed reward measures in the literature. In this work, we defer selection of a reward measure until a POMDP state space has been constructed. The remainder of this sub-section discusses reward function candidates in addition to those listed in section 3.6, which will be explored in this research.

Possible short-term reward measures include:

- Information bit rate: (Seneff and Polifroni, 2000) proposes “Information Bit Rate”, calculated as the average of correct new information shared per turn, as a measure of dialogue quality. This has a natural application as a short-term progress indicator.
- User frustration: (Seneff and Polifroni, 2000) also propose “User frustration” as another metric of dialogue quality. User Frustration is calculated as the number of turns it takes, on average, for an attribute to be successfully transmitted to the system.
- Grounding utilities: (Paek and Horvitz, 2000c) define a formal notion of when something can be said to be grounded. The model relies on costs associated with initiating a repair or tolerating a misunderstanding. Since a POMDP maintains a distribution over user beliefs – in effect, the system’s certainty of user beliefs – a POMDP with the appropriate rewards is effectively implementing this model. (Paek and Horvitz, 1999) shows how these utilities can be derived from psychological experiments. Similarly, (Bohus and Rudnicky, 2001) and (Bohus and Rudnicky, 2002) show how these utilities can be computed from objective analysis of data. The authors seek to identify the relative cost of a false-positive and a false-negative in relationship to objective efficiency measures (e.g., rate of correctly transferred concepts in the dialogue). The authors perform a linear regression of the sum of false-positives and false-negatives with respect to objective efficiency measures to determine the relative cost between the two error types.

In applying costs as short-term reward measures in a POMDP, there is an important question of whether the short-term measures are already accounted for in the long term measures. For example, the work of (Bohus and Rudnicky, 2002) computes immediate costs based on whole-dialogue metrics. It is possible that incorporating these measures as short-term reward measures would be redundant.

Turning to long-term reward measures not fully explored in the literature, PARADISE is a promising candidate. The PARADISE framework (PARAdigm for Dialogue System Evaluation) provides “a decision-theoretic framework to specify the relative contribution of various factors to an agent’s overall performance” (Walker et al., 1998a). In PARADISE, “performance is modelled as a weighted function of a task-based success measure and dialogue-based cost measures.” This naturally lends itself to formulation as a reward measure.

(Walker et al., 1998b) is the only example in the literature of a reward function derived from the PARADISE framework. Yet further work using PARADISE – for example (Kamm et al., 1999) – finds that predictors of user satisfaction are very similar across a variety of domains, including train timetables, email reading, and voice dialling and messaging. (Walker et al., 2000) shows how a predictor of user satisfaction from one domain can well predict user satisfaction in another; however, predictive performance declines when the experience level of the user population changes, or in the face of markedly different levels of ASR performance. PARADISE-derived user-satisfaction prediction scores are promising candidates for long-term POMDP reward functions.

Specification of reward values: Toy POMDP example

In the Toy POMDP, the rewards are given as follows:

- *reward-ask-again* = -3 : reward for asking for a slot after the caller has specified it.
- *reward-confirm-again* = -2 : reward for confirming a slot after the caller has confirmed it.
- *reward-submit-correct* = 10 : reward for taking the action *submit-x-y* where (x,y) matches s_u
- *reward-submit-wrong* = -10 : reward for taking the action *submit-x-y* where (x,y) does not match s_u .
- *reward-fail* = -5 : penalty for taking the *fail* action
- *reward-step* = -1 : per-turn penalty assessed for any action not covered by the above rewards

POMDP assembly: Toy POMDP example

In the Toy POMDP example, the state space is given as the combination of the dialogue state, the user belief state, and the user action:

$$S = \{s_u, s_d, a_u\}$$

A Bayes Net formulation for the full Toy POMDP example is given in Figure 6. Note that diamond nodes indicate *utilities/rewards*, and square nodes represent decisions to be made by the POMDP – i.e., action selection.

To construct a POMDP, we need to formulate the observation function and the transition function explicitly. We create the observation function as follows:

$$\begin{aligned} p(o' | s', a) &= p(o' | s'_u, s'_d, a'_u, a_s) \\ &= p(o' | a'_u) \\ &= p(o | a_u) \end{aligned}$$

This is the ASR corruption model, given above.

We create the transition function as follows:

$$p(s' | s, a) = p(s'_u, s'_d, a'_u | s_u, s_d, a_u, a_s)$$

By standard Bayesian conditional expansion we can write:

$$= p(s'_u | s_u, s_d, a_u, a_s) p(a'_u | s'_u, s_d, a_u, a_s) p(s'_d | a'_u, s'_u, s_d, a_u, a_s)$$

By removing conditional independence, we can write:

$$= p(s'_u | s_u) p(a'_u | s'_u, a_s) p(s'_d | a'_u, s_d, a_s)$$

These terms correspond to the *user belief model*, the *user action model*, and the *dialogue model*, described above.

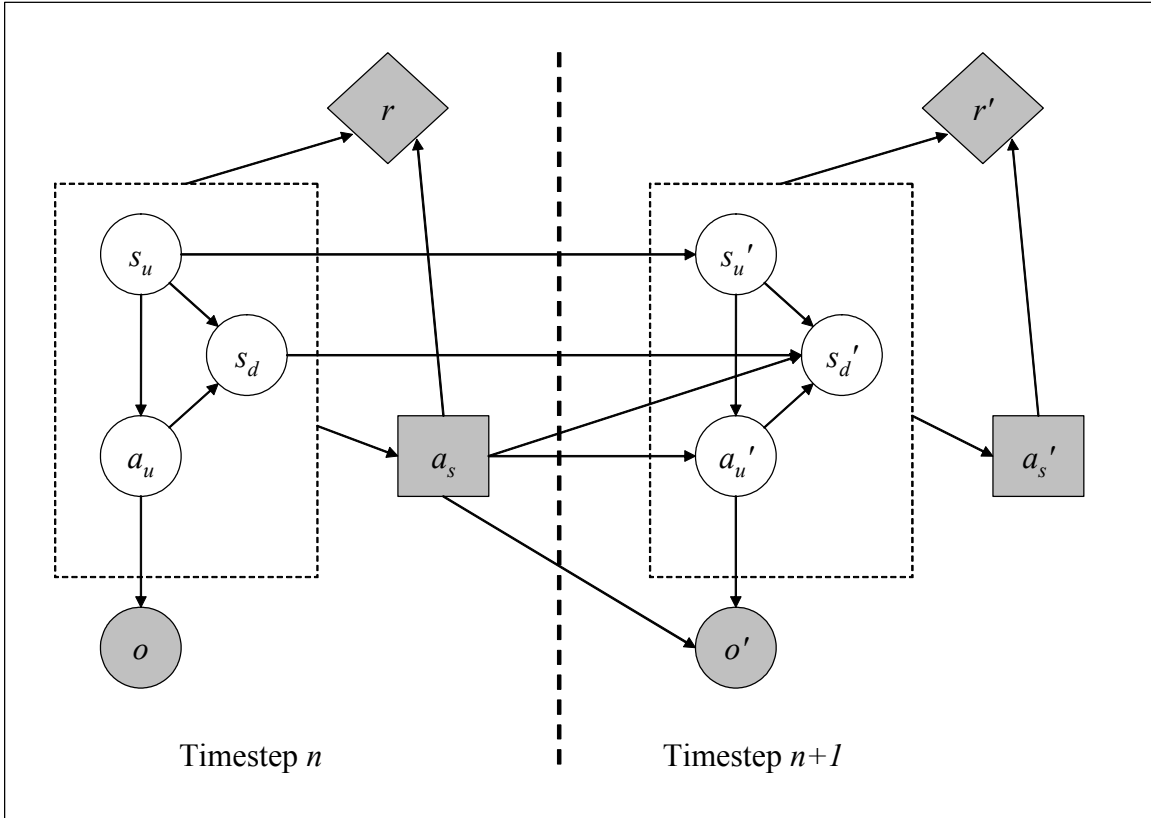


Figure 6: Bayes Net formulation of the Toy POMDP. The dotted rectangles show the extent of the POMDP state space.

Solution methods

Once a complete POMDP model has been constructed, it can be solved for an optimal policy. (Murphy, 2000) provides a good overview of available techniques.

Since we will have a complete model (i.e., one that includes the transition function of the underlying MDP), we have three options: (1) use a heuristic based on the underlying MDP, (2) solve the belief-state MDP using approximate methods, or (3) direct policy search.

Examples of heuristics based on the underlying MDP include:

- Compute the most likely state, and take the optimal action for that state (the MLS approximation)
- Define $Q(b, a) = \sum_{s \in S} b(s)Q^{MDP}(s, a)$ (the Q-MDP approximation)²³
- If the entropy of the current belief state is below a given threshold, use one of the above heuristics; if not, take an action which reduces the entropy (dual mode control).

The problem with heuristics based on the underlying MDP is that they assume that the uncertainty will disappear in the future. These techniques often neglect to perform (effective) information gathering actions. Dual mode control enables the system to more effectively gather information; however, it presents two follow-on problems: how to select a threshold statistic, and how to balance reward with entropy (i.e., we would need to define some trade-off between expected change in entropy and expected total reward).

The first two methods were applied to the POMDP Toy. As expected, the system did not make good use of information-gathering actions. When the entropy was high, the system did take information-gathering actions (because the risk of failure outweighed anything which didn't fail); however the system did not select information-gathering actions intelligently.

The POMDP Toy was further extended to calculate entropy, as well as expected entropy for a given action. By setting an entropy threshold, this system was able to make better use of information-gathering actions. However, as expected, the system sometimes took high-penalty actions to reduce entropy, highlighting the need to define a trade-off between entropy measures and reward measures.

We may alternatively compute a value function over (continuous) belief space; we call this solution approach a Belief-state MDP. Attempting to solve for the value function $V(b)$ exactly is usually intractable. Efficient techniques such as the Witness algorithm (Kaelbling et al., 1998) are based on the realisation that the value function is a piece-wise linear hyperplane in belief space, allowing (at each iteration) fully dominated hyperplanes to be deleted without loss of accuracy. For the POMDP Toy, the Witness algorithm ran for more than 24 hours on a powerful research computer making no progress.

Thus, in practice, POMDP solutions are typically approximate. We can approximate the value function V , approximate b , or approximate both. Since we have an exact model, and computation of b is $O(n^2)$ in number of states, we don't believe that calculating b directly will present a problem. Thus, we will focus on solution techniques which approximate V – for an overview, see (Murphy, 2000).

Recent advancements have shown how even very large POMDPs, when *factored*, can be solved approximately. (Sallans, 2001) claims a good approximate solution to a POMDP with a hidden state space size and an observation space larger than 2^{180} . (Wyatt, 2003) notes that direct policy search is another promising method for solving POMDPs approximately.

²³ $Q(s, a)$ indicates the expected return for taking action a in state s . $Q(b, a)$ indicates the expected return for taking action a in belief state b .

Extensions to the Toy POMDP example

This section has sought to demonstrate that a POMDP can be created from the components proposed in this work. The Toy POMDP has shown basic feasibility of the approach, but will need to be extended in a variety of ways, including:

1. POMDPs use “hard” observations to update the belief state. To integrate a variety of input sources (c.f. section 3.5), it would be desirable to update the state space directly, allowing the incorporation of soft evidence. This might require modifications to the POMDP formulation and to solution methods.
2. The Toy POMDP combined the illocutionary force and propositional content portions of dialogue acts. This work will seek to decompose actions into multiple components.
3. The Toy POMDP assumed a turn consists of exactly one dialogue act. In an actual dialogue, there may be multiple dialogue acts per turn.

4.4 Project proposal, part 4: Optional extension

If time allows, the solved POMDP will be applied to a working system. A further extension would compare two working systems – one with, and one without, the POMDP component – to compare task completion, user satisfaction, etc.

4.5 Project criticisms and risks

Possible project criticisms include:

- The ASR channel simulation could be viewed as an inaccurate representation of a real ASR channel. For example, the introduction of the typist will slow communication more than actual ASR, or the recognition errors may not be representative (perhaps not accounting for prosodic variations which give rise to more misrecognitions).
- A human cannot process *all* available dialogue features (e.g., parse scores, utterance duration, etc.) that a Bayes Net could; thus, a human will perform necessarily worse than a computational dialogue model/manager.

There are several risks with this project, including:

- It’s possible that this work will show that current HH models of grounding and turn-taking are sufficient. This work takes the view that this finding would itself be novel; thus any outcome would be a contribution.
- Recruiting and scheduling participants will be enormously challenging, as will managing the logistics. Extra time has been allocated in the schedule in case subject tests overrun.

- The breadth of activities (data collection, annotation, model construction and evaluation) raises scheduling concerns about completing this work within two years. Extra time has been allocated in the schedule, especially toward later phases of the project, in case phases overrun.

Chapter 5: Project schedule

Table 4 gives a summary of the project schedule. The project is divided into 5 phases: the four phases given in Chapter 4, and a write-up phase.

The schedule has been structured to allow for the possibility of submitting conference papers in March, 2004; as a result, phase 1 is relatively aggressive. If phase 1 runs longer than expected, then no conference papers will be submitted, and the schedule will compensate by curtailing the optional Phase 4.

Phase	Task	Date
1	Prepare for data collection (system implementation and testing, end-pointer and ASR models, recruit subjects)	Sep-Oct 2003
	Data collection ²⁴	Nov 2003
	(Slip for phase 1)	Dec 2003
	PHASE 1 COMPLETE	Dec 31, 2003
2	Transcription (orthographies, dialogue act annotation)	Jan 2003
	Grounding model proposal & evaluation (annotating grounding states and possibly re-annotating dialogue acts)	Feb-Mar 2004
	Optional: Submit conference paper ²⁵	Mar 2004
	(Slip for phase 2)	Apr 2004
	PHASE 2 COMPLETE	Apr 30, 2004
3	Create user model	June 2004
	Iteratively develop POMDP (state space, reward function, solution methods) ²⁶	July-Oct 2004
	(Slip for phase 3)	Nov-Dec 2004
	PHASE 3 COMPLETE	Nov 30, 2004
4	OPTIONAL: Construct a working system ²⁷	Dec 2004 – Feb 2005
Write-up	Write conference papers	Mar 2005
	Write-up, including any final experimentation	Apr 2005-Jun 2005
	(Write-up slip) ²⁸	July-Aug 2005
	SUBMIT	Aug 31, 2005

Table 4: Proposed project schedule

²⁴ Because students will be used as subjects, it will be difficult to perform the subject tests in December.

²⁵ Summer 2004 conference deadlines: ICSLP: Mar 31, 2004; SIGdial: approx. end of March; ACL: 25 Feb 2004; Error handling: (abstract) approx. beginning of March.

²⁶ Some extra time allowed to attend conferences, vacation, etc. during this phase

²⁷ For example, apply to the TALK project.

²⁸ Allows time for summer conferences; could also be used for Journal papers, if thesis material is sufficient

Chapter 6: References

- Alshawi, H. and Douglas, S. (2001). Variant transduction: A method for rapid development of interactive spoken interfaces. *Proc. SIGDial Workshop on Discourse and dialogue*, Aalborg, Denmark..... 13
- Austin, J. (1962). *How to Do Things with Words*. Harvard University Press..... 5
- Balantine, B. (2003). "Principles of Voice User Interface Design." *AVIOS 2003*, San Jose, California... 12
- Bohus, D and Rudnicky, A. (2001). Modeling the Cost of Misunderstandings in the CMU Communicator Dialogue System. *ASRU 2001*, Madonna di Campiglio, Italy..... 34
- Bohus, D. and Rudnicky, A. (2003). RavenClaw: Dialogue Management Using Hierarchical Task Decomposition and an Expectation Agenda. *Eurospeech, Geneva, Switzerland*..... 13
- Bohus, D., and Rudnicky, A. (2002). Integrating Multiple Knowledge Sources for Utterance-Level Confidence Annotation in the CMU Communicator Spoken Dialogue System. Technical Report CS-190, Carnegie Mellon University. 34
- Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistics. *Computational Linguistics*, Vol. 22, No. 2, 249–254..... 14
- Carpenter, P. et al. (1999). Is this conversation on track? Informal technical report. Carnegie Mellon University. <http://www-2.cs.cmu.edu/~15781/web/proj/carpenter.ps> 13
- Chu-Carroll, J and Carpenter, R. (1999). Vector-Based Natural Language Call Routing. *Journal of Computational Linguistics*, Vol. 25, No. 30, 361-388..... 14
- Clark, H. and Brennan, S. (1990). Grounding in Communication. In Resnick et al., editors, *Perspectives on Socially Shared Cognition*. APA..... 5
- Cohen, P. and Perrault, C. (1979). Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, Vol 3, No. 3, 177-212..... 5
- Denecke, M. Rapid Prototyping for Spoken Dialogue Systems. *Proc COLING '02*..... 13
- Doran, C. et al. (2001). Comparing Several Aspects of Human-Computer and Human-Human Dialogues. *Proc 2nd SIGdial Workshop on Discourse and Dialogue*. Aalborg, Denmark.9, 10, 11
- Eckert, W. et al. (1997). User modeling for spoken dialogue system evaluation. Technical Report TR 97.33.1, AT&T Bell Labs Research. 31
- Eckert, W. et al. (1998). Automatic evaluation of spoken dialogue systems. Technical Report TR 98.9.1, AT&T Bell Labs Research.24, 31
- Flammia, G. (1998). *Discourse segmentation of spoken dialogue: an empirical approach*. Ph. D. thesis, MIT..... 6
- Glass, J. (1999). Challenges for Spoken Dialogue Systems. *Proc. 1999 IEEE ASRU Workshop*, Keystone, Colorado, USA. 6, 8
- Goddeau, D. and Pineau, J. (2000). Fast Reinforcement Learning of Dialogue Strategies. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP2000)*. 18

Gorin, A. et al. (1997). How may I help you? <i>Speech Communication</i> , Vol 23, 113-127.	14
He, Y. and Young, S. (2003). A Data-Driven Spoken Language Understanding System. Unpublished Manuscript. Department of Engineering, University of Cambridge, UK.	14
Higashinaka, H. et al. (2003). Corpus-based Discourse Understanding in Spoken Dialogue Systems. <i>Proc 41st Annual Meeting of the Association for Computational Linguistics</i> . Sapporo, Japan.	15
Hirschberg, J. et al. (2001). Identifying User Corrections Automatically in Spoken Dialogue Systems. <i>Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'01)</i> , Pittsburgh, PA.	13
Horvitz, E. and T. Paek. (2000). DeepListener: Harnessing Expected Utility to Guide Clarification Dialogue in Spoken Language Systems. <i>6th International Conference on Spoken Language Processing (ICSLP 2000)</i> , Beijing.	16
Jensen, F. (2001). <i>Bayesian Networks and Decision Graphs</i> . Springer Verlag, New York.	30
Kaelbling, L. et al. (1998). Planning and Acting in Partially Observable Stochastic Domains. <i>Artificial Intelligence</i> , Vol. 101.	37
Kamm, C., et al. (1999). Evaluating Spoken Language Systems. <i>Proc American Voice Input/Output Society (AVIOS)</i>	34
Krahmer, E. et al. (1999). Problem Spotting in Human-Machine Interaction, 1423-1426. <i>Proc. Eurospeech</i>	13
Krahmer, E. et al. (2001). Error Detection in Spoken Human-Machine Interaction. <i>International Journal of Speech Technology</i> , Vol 4, No. 1, 4, 19-30.	13
Langkilde, I., et al. (1999). Automatic Prediction of Problematic Human-Computer Dialogues in "How May I Help You?" <i>Proc Automatic Speech Recognition and Understanding Workshop (ASRU'99)</i> , 369-372. Keystone, Colorado, USA.	14
Larsson, S. (1998). Coding Schemas for Dialogue Moves, Technical report. Department of Linguistics, Goteborg University.	28
Larsson, S. and Traum, D. (2000). Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. <i>Natural Language Engineering, special issue on Best Practice in Dialogue Systems Design</i>	13
Lemon, O. et al. (2002b) Multi-tasking and Collaborative Activities in Dialogue Systems. <i>Proc of 3rd SIGdial Workshop on Discourse and Dialogue</i>	13
Levin et al. (2000). A Stochastic Model of Human-Machine Interaction for Learning Dialogue Strategies. <i>IEEE Transactions on Speech and Audio Processing</i> , Volume 8, No. 1, 11-23.	17
Levin, E. and Pieraccini, R. (1997). A Stochastic Model of Computer-Human Interaction For Learning Dialogue Strategies. <i>Proc Eurospeech</i> , Rhodes, Greece.	17
Levin, E. et al. (1998). Using Markov Decision Processes For Learning Dialogue Strategies. <i>Proc Int Conf Acoustics, Speech and Signal Processing</i> , Seattle, USA.	17, 19
Levinson, S. (1983). <i>Pragmatics</i> . Cambridge University Press.	6
Levoa, G.-A. (2003). Learning to Speak to a Spoken Language System: Vocabulary Convergence in Novice Users. <i>Proc 4th SIGdial Workshop on Discourse and Dialogue</i> . Sapporo, Japan.	11
Levow, G.-A. (1998). Characterizing and recognizing spoken corrections in human-computer dialogue. <i>Proc COLING-ACL</i> . Montreal, Canada.	10
Lewin, I. (1998). The autoroute dialogue demonstrator. Tech. rept. CRC-073. SRI Cambridge Computer Science Research Centre.	14
Litman, D. et al. (2000). Predicting automatic speech recognition performance using prosodic cues. <i>Proc. 1st Conference of the North American Chapter of the Association for Computational Linguistics</i>	10

Litman, D. et al. (2001). Predicting User Reactions to System Error. <i>Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL'01)</i> , Toulouse, France.	13
Litman, D. et al. (1998). Evaluating Response Strategies in a Web-Based Spoken Dialogue Agent. <i>Proc 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98)</i> , 780-786. Montreal, Canada.	8
Litman, D., and Pan, S. (2000). Predicting and Adapting to Poor Speech Recognition in a Spoken Dialogue System. <i>Proc 17th National Conference on Artificial Intelligence (AAAI-2000)</i> , 722-728. Austin, Texas, USA.	13
McInnes, F. et. al. (1999). Effects of prompt style on user responses to an automated banking service using word-spotting. <i>BT Technical Journal</i> , Vol. 17, No. 1, 160-171.	26
Moore, R. and Browning, S. (1992). Results of an exercise to collect "genuine" spoken enquiries using woz techniques. <i>Proceedings of the Institute of Acoustics</i> , Vol 14, No. 6.	9
Muller, C. and Strube, M. (2003). Multi-Level Annotation in MMAX. <i>Proc 4th SIGdial Workshop on Discourse and Dialogue</i> . Sapporo, Japan.	28
Murphy, K. (2000). A survey of POMDP solution techniques. Informal technical report. Massachusetts Institute of Technology. http://www.ai.mit.edu/~murphyk/Papers/pomdp.ps.gz	36
Ng, H.-I. and Lua, K.-T. (2003). Dialogue Input Ranking in a Multi-Domain Environment Using Transferable Belief Model. <i>Proc 4th SIGdial Workshop on Discourse and Dialogue</i> . Sapporo, Japan.	14
Orasan, C. (2003). PALinkA: A highly customisable tool for discourse annotation. <i>Proc 4th SIGdial Workshop on Discourse and Dialogue</i> . Sapporo, Japan.	28
Paek et al. (2000). Continuous Listening for Unconstrained Spoken Dialogue. <i>ICSLP 2000</i> , Beijing.	16
Paek, T. and Horvitz, E. (1999). Uncertainty, Utility, and Misunderstanding: A Decision-Theoretic Perspective on Grounding in Conversational Systems. <i>AAAI Fall Symposium on Psychological Models of Communication</i> , pp 85-92. North Falmouth, MA.	33
Paek, T. and Horvitz, E. (2000a). Conversation as Action Under Uncertainty. <i>Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)</i> , Stanford, CA.	15
Paek, T. and Horvitz, E. (2000c). Grounding Criterion: Toward a Formal Theory of Grounding. Technical Report MSR-TR-2000-40. Redmond, Washington: Microsoft Research.	33
Pietquin, O. and Renals, S. (2002). ASR Modelling for Automatic Evaluation and Optimisation of Dialogue Systems. <i>Int Conf Acoustics Speech and Signal Processing</i> , Florida.	18, 20
Pirker, H. and Loderer G. (1999). "I said TWO TI-CKETS": How to Talk to a Deaf Wizard. <i>Proc ESCA Workshop on Dialogue and Prosody</i> , Veldhoven, The Netherlands.	10
Roy, N. et al. (2000). Spoken Dialogue Management Using Probabilistic Reasoning. <i>Proceedings of the ACL 2000</i>	21
Sallans, B. (2001). <i>Reinforcement Learning for Factored Markov Decision Processes</i> . Ph. D. thesis, Dept. of Computer Science, University of Toronto.	37
Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. <i>HLT 2002</i> , San Diego, USA.	18, 19, 20
Searle, J. (1976). A classification of illocutionary acts. <i>Language in Society</i> , Vol 5, 1-23.	5
Seneff, S. and Polifroni, J. (2000). Dialogue Management in the Mercury Flight Reservation System. <i>Satellite Dialogue Workshop, ANLP-NAACL</i> , Seattle, WA.	13
Sheeder, T. and Balough, J. (2003). Say it Like You Mean It: Priming for Structure in Caller Responses to a Spoken Dialogue System. <i>International Journal of Speech Technology</i> , No. 6, 103-111.	11
Shin, J. et al. (2002). Analysis of User Behavior under Error Conditions in Spoken Dialogue. <i>ICSLP</i>	11

- Singh, S. et al. (2002). Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence*, Vol. 16, 105-133.17, 19
- Stolcke, A., et al. (2000). Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, Vol 26, No. 3, 339-373. 14
- Sutton, R. and Barto, A. (1998). *Reinforcement Learning: an Introduction*. The MIT Press. Cambridge, Massachusetts, USA.16, 19
- Traum, D. (1994). *A Computational Theory of Grounding in Natural Language Conversation*. Ph. D. thesis, Department of Computer Science, University of Rochester, Rochester, NY. 5
- Traum, D. (2000). 20 Questions for Dialogue Act Taxonomies. *Journal of Semantics*, Vol. 17, 7-30. 14
- Walker et al. (2000). Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, Vol. 6, No. 3. 34
- Walker, M. et al. (1998a). Evaluating Spoken Dialogue Agents with PARADISE: Two Case Studies. *Computer Speech and Language*, Vol. 12, No. 3. 34
- Walker, M. et al. (1998b). Learning Optimal Dialogue Strategies: A Case Study of a Spoken Dialogue Agent for Email. *Proceedings of ACL/COLING 98*.17, 19, 34
- Wyatt, J. (2003). An introduction to Reinforcement Learning (Part 2). *Proc. BCS-PAR Summer School on Pattern Recognition, Exeter University, Exeter, UK*. 37
- Zhang, B. et al. (2001). Spoken Dialogue Management as Planning and Acting under Uncertainty. *Proc Eurospeech*. Aalborg, Denmark. 21
- Zue, V. (1997). Conversational Interfaces: Advances and Challenges. *Proc Eurospeech*. 6