Reasoning about interaction in mixed-initiative artificial intelligence systems

by

Michael William Fleming

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Computer Science

Waterloo, Ontario, Canada, 2003

UW TR CS-2007-15

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis presents computational models for the design of mixed-initiative artificial intelligence systems that can make rational decisions about interaction with potentially helpful users. Mixed-initiative systems are ones in which either the system or the user can take the initiative to direct the dialogue or the problem solving. These systems have been designed for such diverse applications as robotics, military planning, intelligent tutoring and trip scheduling. One challenge in designing these systems is to specify when the system should take the initiative to interact with the user. The main contribution of the thesis is to provide designers of mixed-initiative systems with a systematic approach for constructing systems that can reason in a principled way about interaction with the user, regardless of the area of application. Our approach is to model the user, the task and the dialogue simultaneously. Specific factors are proposed that must be modeled, and methods are developed for how to combine these factors in order to make rational decisions about interaction, based on whether the perceived benefits of communication exceed the expected costs. Some examples and experiments are described, to demonstrate the value of the models and to justify decisions that were made in determining the role of each factor in the computational models. In particular, we emphasize the value of making decisions about interaction based on a careful evaluation of the needs, preferences and abilities of the user, leading to mixed-initiative systems that are user-specific and therefore result in greater overall user satisfaction.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Robin Cohen. If ever there was a supervisor who epitomized the phrase "going beyond the call of duty," it would be you, Robin. I know that I was not the easiest student to supervise, and I appreciate all your help and guidance over the *many* years I have spent at Waterloo. Thank you very much!

Thanks also to the members of my thesis examining committee: Dr. Dale Schuurmans, Dr. Frank Tompa, Dr. Olga Vechtomova and Dr. Ingrid Zukerman. I greatly appreciated your constructive criticism, your many helpful suggestions (and, most of all, the fact that you approved the thesis!). I would also like to acknowledge the many people I have spoken to at conferences who have provided helpful suggestions for my work, including Mathias Bauer, Jim Greer, Tony Jameson, Neal Lesh, Diane Litman, Gord McCalla, Robert St. Amant and Candy Sidner.

I would like to thank the support staff (past and present) in the UW Computer Science department – in particular, Wendy Rush and Jane Prime.

Thanks also to Dr. Jane Fritz, Dr. Virendra Bhavsar and everyone else at the University of New Brunswick for providing me with the opportunity to begin my career as a professor while I was still completing my thesis at Waterloo. The atmosphere at UNB has been fantastic and has made the transition easier than I ever could have expected.

I would like to acknowledge the University of Waterloo, the Natural Sciences and Engineering Research Council and PRECARN/IRIS for their financial support during my time as a Ph.D. student.

A big thank you goes out to my family – to Mom and Dad, to David, Christiana, Bailey, Amelia and Sophie, to the McCurdys and to all of my extended family. I've really appreciated all of your love and support.

Finally, Wendy... None of this would have been possible without your love, encouragement and tolerance over the years. Hard to believe that it is finally finished! Thank you so much for everything! This thesis is dedicated to you and to my biggest inspirations over the last couple of years, Benjamin and Hannah. I love all of you!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Research in artificial intelligence aims to produce computer systems with the ability to make intelligent decisions – and with some amount of autonomy to make such decisions. In designing artificial intelligence systems to operate autonomously, the view of processing is generally as follows. The user first provides some input. The system makes use of this input as it reasons autonomously and then returns its output to the user. The topic of mixed-initiative systems, which has become a subject of much discussion in the AI research community (Haller and McRoy, 1997; Cox, 1999; Haller, Kobsa and McRoy, 1999), suggests a more flexible approach to the reasoning process.

In a mixed-initiative system, both the computer and the user can play an active role in a problem-solving session. At any given time, either party might take control of a session. The primary goal behind a mixed-initiative system is to take advantage of the fact that computers and people have very different strengths when it comes to solving problems. While machines excel at performing calculations quickly and

searching through large search spaces, humans often have a better intuitive sense of how to solve a problem and a better feel for the higher-level goals of what they are trying to achieve.

It is important to note that, in a true mixed-initiative system, the roles of the two parties are not determined ahead of time. Control of the task or dialogue might pass from one participant to the other frequently within a session, sometimes in quite unpredictable ways (Chu-Carroll and Brown, 1997; Allen, 1994; Burstein and McDermott, 1996). A well-designed system should be able to adjust to each situation in an appropriate way: to take control when its own abilities are better suited to a particular aspect of the task and to yield to the user when the latter's abilities and knowledge are deemed to be superior.

Several workshops and symposia have been held in recent years to bring together researchers interested in mixed-initiative systems. In 1997, there was a AAAI spring symposium on *Computational Models for Mixed Initiative Interactions* (Haller and McRoy, 1997). A 1998 AAAI spring symposium focused on *Interactive and Mixed-Initiative Decision-Theoretic Systems* (Haddawy and Hanks, 1998). In 1998-99, the journal *User Modeling and User-Adapted Interaction* released a special double issue on *Computational Models for Mixed Initiative Interaction*, which was later published as a book (Haller, Kobsa and McRoy, 1999). In 1999, AAAI held a workshop on *Mixed-Initiative Intelligence* (Cox, 1999). A workshop entitled *Mixed-Initiative Intelligent Systems* took place at the IJCAI conference in Acapulco in August 2003.

Much of the existing work on mixed-initiative systems has been divided be-

tween two main groups. One group involves natural language researchers focused on mixed-initiative dialogue models and on managing the interaction which must take place when multiple parties are working together on a task. The second group is interested in looking at specific problem domains, and the ways in which mixed-initiative techniques can lead to improvements over existing automated systems. The most notable subgroup here is an offshoot of the AI planning community, concerned with investigating the introduction of mixed-initiative techniques into planning systems (Allen, 1994; Burstein and McDermott, 1996; Myers, 1998), and the relevant issues which arise. Existing work from these two groups will be discussed in some detail in Chapter 2.

One ideal feature of a mixed-initiative system is the ability to make intelligent decisions about *whether or not* it is appropriate to seek further assistance from a user at any given time. In some situations, there might be no debate at all: without additional input from the user, there is no way for the system to proceed with the task at hand. For example, in the domain of planning a travel itinerary for a user, it is unreasonable for the system to be expected to suggest appropriate flights to the user without being aware of the departure and destination cities. In other situations, there might be no advantage at all to asking the user for assistance. Again, in the travel domain, if the system knows of a flight that satisfies *all* of the user's preferences, then there is no need for it to ask the user to indicate which preference is more important.

The interesting cases arise when a system perceives that there might be some advantage to obtaining more guidance from the user, but when there is some cost

associated with this interaction as well. It is this observation that has motivated our research into the problem of providing a principled approach to reasoning about interacting with potentially helpful users. We propose that a system should solicit further input from a user precisely when the perceived benefits of this interaction exceed the expected costs.

In this thesis, a model will be presented for how to design a system that can make exactly this type of decision. With this model clearly defined, it would then be possible to design mixed-initiative systems, for any application area, that clearly stipulate when the system should take the initiative to interact with the user.

1.1 Problem statement

The central problem addressed in this thesis can be summarized as follows:

- Artificial intelligence systems are often faced with situations in which their ability to perform a task (or to collaborate with users on a task) could be improved by soliciting further information from potentially helpful users. However, such user interactions also have costs associated with them.

- What is needed, and has been lacking in the literature so far, is a principled and general method for systems to use in reasoning about whether or not to initiate information-seeking dialogues with users. This decision should be made by analyzing the benefits and costs of interactions with users.

Our principled solution to this problem will be dependent on the idea of modeling the user, the task and the dialogue simultaneously. We propose specific factors

that must be modeled and mechanisms for how to combine these factors in order to make rational decisions about interaction. This solution is best viewed as a decision-theoretic approach, in which the benefits of interaction are weighed against the perceived costs. The user model plays a critical role in this decision process. Decisions about interaction must be made not only on the basis of the current state of the problem solving, but also on whether, with *this user*, it is worthwhile to interact and whether, *at this point in the dialogue*, interaction is likely to be successful.

1.2 Organization

This thesis is organized as follows. In Chapter 2, some details are provided on background research in mixed-initiative interaction, user modeling and intelligent user interfaces. In Chapter 3, several models are introduced specifying how a system can make rational decisions about whether or not to interact with a user in a given situation. A concrete decision procedure is provided to guide system designers in determining which of these models is appropriate for their application domain. In Chapter 4, a few examples are described, to illustrate the decision procedure and the models for different application areas. Some experiments are described in Chapter 5, to help to validate the models. In Chapter 6, detailed discussion is provided, including a summary of the significant contributions of this research and a list of extensions that were not discussed in the core presentation of the model in Chapter 3. Related work is presented in Chapter 7, while Chapter 8 contains a summary of the research and a list of possible topics for future work.

Chapter 2

Background

In this chapter, some background work will be presented from two main subareas of artificial intelligence research: mixed-initiative interaction and user modeling. Throughout this discussion, we will clarify the important background concepts that are central to the development of the problem-solving models in this thesis and we will outline the main areas of research to which this thesis makes a contribution.

2.1 Mixed-Initiative Systems

As discussed in Chapter 1, much of the existing work on mixed-initiative systems has been divided between two main subgroups. One group involves natural language researchers focused on mixed-initiative dialogue models and on managing the interaction which must take place when multiple parties are working together on a task. The second group is interested in looking at specific problem domains, and the ways in which mixed-initiative techniques can lead to improvements over

existing systems. The most notable subgroup here is an offshoot of the AI planning community, concerned with investigating the introduction of mixed-initiative techniques into planning systems (Allen, 1994; Burstein and McDermott, 1996; Myers, 1998), and the relevant issues which arise.

2.1.1 Mixed-Initiative dialogue

One of the first papers to introduce the term "mixed-initiative" was by Walker and Whittaker (1990). In this work, mixed-initiative human-human dialogue is modeled by examining the different methods by which control can be transferred from one conversant to another. The paper classifies natural language utterances into four groups, and identifies whether or not the speaker or hearer has control after each type of utterance has occurred:

- assertion: speaker control, unless response to a question
- command: speaker control
- question: speaker control, unless response to a question or command
- prompt: hearer control

Walker and Whittaker (1990) also discuss *interruptions* in dialogues, and investigate *why* dialogue participants might decide to interrupt a speaker. They present two possibilities. If a listener interrupts because he is unsure about the truth of a speaker's statement or finds it ambiguous, he is said to have a problem with *information quality*. On the other hand, an interruption occurs because of *plan quality*

concerns when the listener believes that the speaker's proposed goal is unclear or presents an obstacle.

This distinction is somewhat related to the idea of separating dialogue and task initiative, as presented by Chu-Carroll and Brown (1997). Prior to Chu-Carroll and Brown's work, researchers discussed the construction of mixed-initiative systems (e.g., (Walker and Whittaker, 1990; Smith and Hipp, 1994; Guinn, 1996)) without making this distinction. Dialogue initiative involves control of the *flow of conversation* by a participant, while task initiative refers to the control of the *current problem-solving process*, with both participants working together to achieve some goal. Chu-Carroll and Brown suggest that systems that model only a single thread of control fail to account for the important difference between the case in which a conversant is simply pointing out a flaw in the other conversant's plans, and the case in which the conversant is actively suggesting an alternative course of action.

In Figure 2.1 (from Chu-Carroll and Brown (1997)), utterances (3b) and (3c) illustrate this difference. In (3b), agent A is taking over only the dialogue initiative, while in (3c), it has taken both the dialogue initiative and task initiative.

Even though there is still discussion within the field about the most appropriate definition for initiative (Novick and Sutton, 1997; Cohen et al., 1998), many systems are being built and labelled as mixed-initiative systems, as long as they allow the user and system to take the initiative to control either the dialogue or the problem solving during the processing.

(1) S: I want to take NLP to satisfy my seminar course requirement.

(2) Who is teaching NLP?

(3a) A: Dr. Smith is teaching NLP.

(3b) A: You can't take NLP this semester because you haven't taken AI, which is a prerequisite for the course.

(3c) A: You can't take NLP this semester because you haven't taken AI, which is a prerequisite for the course.

I would suggest that you take some other seminar course to satisfy your requirement, and sign us as a listener for NLP if you're really interested in it.

Figure 2.1: Illustrating dialogue and task initiative

More recently, Walker, Litman and their colleagues (Walker et al., 1997b) have designed the PARADISE system, a general framework for evaluating spoken dialogue agents. The basic idea is to measure task success and dialogue cost, under the assumption that maximizing the former and minimizing the latter will lead to a high level of user satisfaction.

One of the applications of this framework has been to empirically determine the value of a mixed-initiative dialogue strategy, as compared to a system-initiative design (Walker et al., 1997a). Their system-initiative strategy "acquires information in small increments and constrains the user to single word utterances whose content is explicitly prompted for," while the mixed-initiative strategy "lets the user control the dialog, doesn't provide information unless the user asks for it, and allows the user random access to all the application functionality with utterances that combine a set of information requirements."

Figure 2.2 shows examples from ELVIS (EmaiL Voice Interactive System) to illustrate the difference between the two approaches (Walker et al., 1997a). In this paper, experiments are described in which users are asked to perform three simple tasks involving retrieving information from e-mail messages via the telephone. In these experiments, the system-initiative strategy is actually found to be superior to the mixed-initiative strategy, based on the performance function derived using the PARADISE framework. The authors speculate that there are two main causes for this (perhaps counter-intuitive) finding. First, current speech recognizers are not perfect, and recognition errors are more common with the mixed-initiative strategy, since the user is free to use a less restricted vocabulary. The second problem is that users of the mixed-initiative strategy are often unsure of what they should say at any given moment, since they are not aware of the system's capabilities and limitations.

Walker et al. (1997a) propose that the mixed-initiative strategy is a more appropriate technique for *experienced* users. This is supported by their finding that, on the last of the three tasks in the experiment, users of the mixed-initiative strategy reported higher confidence when asked if they knew what to say to the agent. General user satisfaction also improved over time.

More recent work from this group has moved toward the design of systems that can *adapt* their dialogue strategies during the course of the dialogue. The system can shift among system-initiative, user-initiative and mixed-initiative strategies, and can also confirm what it has heard from a user's utterance explicitly, implicitly, or not at all. In a paper by Litman and Pan (1999), it was shown that the standard

(SI) A: Hi, Elvis here. You have 5 new and 0 unread messages in your inbox. Say Repeat to repeat this message, or say Read, Summarize, or I'm done here.

U: Read.

A: Select by Content or Position?

U: Content.

A: Select by Sender or Subject?

U: Sender

A: Which Sender?

U: Kim.

A: The message from Kim is about Meeting Tomorrow. The meeting tomorrow is at 10:30 in 2D-516.

(MI) A: Hi, Elvis here. I've got your mail.

U: Read me my messages from Kim.

A: The message from Kim is about Meeting Tomorrow. The meeting tomorrow is at 10:30 in 2D-516.

Figure 2.2: System-initiative and mixed-initiative dialogue strategies

version of a train schedule retrieval system was outperformed by an adaptable version in which users were able to adjust the initiative and confirmation strategies. In a follow-up paper by Litman and Pan (2000), the *system* uses learning techniques to anticipate speech recognition problems and to adjust the dialogue strategy accordingly. This system also achieves improved performance over a non-adaptive system. In particular, the task success rate on a set of train schedule retrieval tasks improved from 23% to 65%.

However, the overall approach in these papers is still to have the system behave according to one of a fixed set of dialogue strategies, rather than treating each individual situation separately. In contrast, this thesis will present instead a strategy for deciding whether to interact, at each decision point in the process.

2.1.2 Mixed-initiative Planning

The general goal of research in mixed-initiative planning is to develop systems that will allow two or more planning agents (humans or computers) to work together toward solving artificial intelligence planning problems: problems that involve the construction of sequences of actions that will achieve a set of goals.

Throughout the planning process, there is a need for interaction between the planning agents. The ideal situation would be one in which the interaction and co-ordination of activities are not specified in advance, but are instead adapted to the particular problem being solved and to the participants involved. Control of the dialogue and of the task execution could flexibly pass from one party to another, taking advantage of the specific strengths of the individual planning agents.

Two specific position papers provide an in-depth investigation of mixed-initiative planning issues (Allen, 1994; Burstein and McDermott, 1996). The latter states that the "overall objective of research on mixed-initiative planning (MIP) is to explore productive syntheses of the complementary strengths of both humans and machines to build effective plans more quickly and with greater reliability." It goes on to identify those complementary strengths. While humans are "better at formulating the planning tasks," machines are "better at systematic searches of the spaces of possible plans."

Both of these papers present a number of issues that arise in designing mixed-initiative planning systems. Among these are the following points:

- specification of **when** exactly the system and user should communicate, and what that communication should look like;
- registration of context when one party interrupts the other;
- ensuring that both parties share the responsibility involved in the task, and are fully aware of the responsibilities of each party.

These issues and others are identified, but not resolved in these early papers. These are crucial concerns, not only in the design of mixed-initiative *planning* systems, but in any system using mixed-initiative techniques.

In addition to these mostly theoretical papers, there has been some work on implementing specific mixed-initiative planning systems. One of the most notable of these is the TRAINS project from the University of Rochester (Ferguson, Allen and Miller, 1996). The aim of this endeavour is to produce a fully integrated planning

system that is capable of understanding spoken and typed natural language, and of managing interaction with a user through a graphical user interface. The application domain involved has been that of scheduling transportation in a railroad freight system. A later system, *TRIPS: The Rochester Interactive Planning System* (Ferguson and Allen, 1998), attempts to incorporate more realistic and more complex planning tasks in the logistics domain. Figure 2.3 shows an actual dialogue from the TRAINS-95 system (Ferguson, Allen and Miller, 1996). This work established a dialogue-based approach as an effective technique for interactive planning and emphasized the need for communication between automated and human planners in an environment in which neither would be very successful alone.

Another relevant project is Cox and Veloso's work on case-based military force deployment planning (Cox and Veloso, 1997; Veloso, Mulvehill and Cox, 1997). Much of the focus of this research has been on investigating, and attempting to overcome, the differences between the way automated planners work and the manner in which human planners tend to think about problems. In particular, humans tend to think of goals in terms of actions that need to be performed, rather than in terms of the state-based representation used by many planners. This problem is solved by having the system use a preprocessing step to translate actions into states before execution. Users also tend to provide both top-level goals and subgoals, whereas most automated planners expect only top-level goals. This is addressed by a set of control rules that "prefer top-level goals before lower-level goals and then prefer bindings for operators that opportunistically solve user-provided subgoals" (Cox and Veloso, 1997).

```
1.    M:  I need to get to Detroit, Syracuse, and Montreal.
2.    S:  Ok.  [highlights the three cities]
3.    M:  Let's send the trains from Chicago to Detroit.
4.    S:  Ok.  [draws route from Chicago to Detroit via Toledo]
          But the terminal at Toledo is delaying traffic due to
          bad weather.  Trains will take an additional five hours
          to move through there.
5.    M:  That's ok.  Now let's go from Atlanta up to Syracuse.
6.    S:  Please tell me a route to use to get from Atlanta to
          Syracuse.  [highlight engine at Atlanta and city Syracuse]
7.    M:  Go via Charleston and Pittsburgh.
8.    S:  Ok.  [draws route via Cincinnati] But traffic through
          Cincinnati is being delayed due to construction.
9.    M:  OK, let's go via Charleston and Buffalo instead.
10.   S:  No problem.  [draws route]
11.   M:  That's great.
12.   S:  I hope it was helpful.
13.   M:  Finally, I want to go to Montreal from Philadelphia.
14.   S:  OK. [draws route]
15.   M:  That's great.  I'm done.
```

Figure 2.3: Example dialogue from TRAINS-95

Myers (2000) has been investigating the usability of planning systems and has proposed that it is necessary for automated planners to be designed so that the user is able to take some control of the planning process and to understand any progress that the planning agent might be making on the problem. She develops a planning model that accomplishes these goals by including three "user-centric" planning capabilities: *user directability of planning*, *generation of qualitatively different plans*, and *plan summarization*.

Myers and Morley (2001) describe a prototype multiagent system that uses a framework built on this idea of agent directability. They point out that most previous research on planning systems has focused on the automation of planning tasks and on techniques for communication between agents. Meanwhile, very little work has been done on allowing human users, who often wish to play an active role in the planning process, to be involved. They advocate mixed-initiative control of the planning process, and they emphasize the need for human planners to be able to specify goals in a high-level language and to view the status of an agent's progress on its assigned tasks.

Recently, this work has evolved into the PASSAT (Plan-Authoring System based on Sketches, Advice, and Templates) framework (Myers et al., 2001; Myers et al., 2003). This system includes a mixed-initiative plan sketch facility, allowing the human and agent to work together on incrementally refining plan outlines until a satisfactory solution is reached.

2.1.3 Other domains

Intelligent Tutoring Systems

Some work has also been done on mixed-initiative intelligent tutoring systems. For example, Aist (1997) has looked at deciding when a task is complete and when to intervene and provide assistance in an oral reading tutor, Carberry (1997) has used the field of medical training to examine the consequences of allowing users to take a more active role in the learning process, and Lester et al. (1997) have looked at mixed-initiative problem solving using animated pedagogical agents.

Intelligent tutoring is one application that certainly seems to lend itself well to mixed-initiative techniques. There are situations in which a tutor wishes to guide the student along, presenting facts and examples, and asking specific questions to test how well the student has understood the material being introduced. On the other hand, it is often helpful for a tutor to step back and allow the student to guide his own learning process, with the tutor interjecting only when the student is in need of assistance or could benefit from some additional information which he is not likely to discover on his own.

Intelligent tutoring turns out to be a particularly interesting application, since certain issues arise in this domain that are not present in many others.

- With intelligent tutoring, the goal is not just to accomplish the task currently in focus, but also includes a desire for one of the parties (the student) to *learn*, to acquire knowledge that could be applied to future tasks as well.

- There can be some ambiguity as to whether or not a task is complete. While

task completion can be quite clear in some domains, it is difficult to determine in a tutoring dialogue when the student has satisfactorily learned the current material. With this ambiguity, questions can arise as to which party should be able to propose (or unilaterally decide) that the task has been completed.

- Some ITS researchers (*e.g.*, (Carberry, 1997; Keim, Fulkerson and Biermann, 1997)) have proposed that there should be a connection between the level of expertise of the student and the amount of control he should be given over the task.

- Many factors must be considered when determining how to interpret silence on the user's part, and how long to wait before intervening (Aist, 1997). This includes information about the particular student, the last action taken by the student and the tutor, and the difficulty of the current task.

Robotics

Kortenkamp et al. (1997) describe the use of mixed initiative in the application of "traded control" with an autonomous robot. This involves situations in which a robot is sometimes autonomous, but is controlled by a human at other times. Dangerous errors can occur when the human yields control back to the robot. They describe a software architecture designed to deal with both robot control and human control; this architecture includes a mixed-initiative planner. At the recent AAAI Mobile Robot Competition panel at the AAAI-02 conference, it was mentioned that robotics generally is moving to mixed-initiative designs for their

systems.[1]

Other domains

Other research on the use of mixed-initiative systems has included work by Jackson and Havens (1995) on mixed-initiative solving of constraint satisfaction problems, by Tecuci et al. (1999) on mixed-initiative development of knowledge bases, and by Rich and Sidner (1997a) on Collagen, an application-independent toolkit for developing collaborative interface agents. Among other applications, the Collagen project has included work on a mixed-initiative trip planning system. Collagen will be discussed in more detail in Section 2.3.

2.1.4 Research Challenges for Mixed-Initiative Systems

The research described above demonstrates the range of applications for which a mixed-initiative approach has been considered appropriate. Some researchers have been investigating how to model initiative and how to manage the interaction in a mixed-initiative dialogue; others have been developing specific systems that attempt to take the initiative and to relinquish it at appropriate times, according to the perceived strengths of the participants.

Essentially, however, all of these projects have been independent; what is needed is a general, unifying framework for how to design mixed-initiative systems, one that models both the problem solving and the underlying dialogue.

[1] The robot competition, in fact, featured a robot, Grace, that asked for directions from conference attendees in order to find the registration desk.

2.2 User Modeling

The models presented in this thesis will rely quite heavily on the idea of a user model. User modeling is the representation of any information about a user that might influence the system's behaviour: her knowledge, preferences, beliefs, abilities, goals and plans. This information is represented in a separate knowledge base and is used by the system to adapt its behaviour to each individual user (Kobsa and Wahlster, 1989). This allows the system, for example, to adjust the presentation of information to suit a user's needs (Paris, 1991) or to better assist a user by taking into account her abilities with respect to different aspects of a task (Chin, 1989).

User modeling techniques are evident in a wide range of application areas. Recommender systems (Kautz, 1998) suggest movies, music, books and other products to users according to the profiles that they have built up for those users. Intelligent tutoring systems adjust the manner in which tutorial information is presented, according to information stored about the user's expertise and about her previous sessions with the system (Kass, 1989). Lieberman (1995) describes a system that recommends web sites based on the user's past viewing habits.

There are two main approaches to the acquisition of user modeling information: explicit and implicit (Konstan et al., 1997; Billsus and Pazzani, 1999). In explicit user model acquisition, information is provided directly by the user or is obtained by asking her specific questions about his preferences, goals or knowledge. The answers to these questions are then used by the system in determining how to adjust its behaviour to each individual user. For example, a user might be asked to rate her knowledge of the Unix operating system as novice, beginner, intermediate

or expert (Chin, 1989). The system would then adjust its tendency to provide help to the user according to this explicit information.

Implicit user model acquisition involves a more subtle approach to collecting data. The system simply observes the user as he uses an application and records relevant information. For example, if it is observed that a user has purchased the music of a certain artist in the past, then the system might conclude that the user would be very interested in a newly-released CD from that artist. Implicit techniques often use heuristics (Kass, 1991) in order to draw conclusions from the patterns that are detected.

It is in fact quite common to use a combination of explicit and implicit techniques: for example, using explicit acquisition to get an initial rough model of the user, and then using implicit techniques to refine this model without bothering the user any further (Kass and Finin, 1988).

Another important concept in user modeling research is that of a stereotype. "A stereotype represents a collection of attributes that often co-occur in people" (Rich, 1989). The idea is to use this type of information to allow systems to predict the preferences or knowledge of a user based on general characteristics of a class of users to which this user is believed to belong. In the Unix example mentioned above (Chin, 1989), the system would adjust its behaviour according to whether the user has identified himself as an expert, an intermediate user, a beginner or a novice. Ardissono and Goy (2000) describe the use of stereotypes to personalize the presentation of products in a web store.

Stereotypes are used as a mechanism to infer user modeling information about

users, without the benefit of explicit acquisition or of implicitly inferring from the user's individual actions. They are particularly useful in situations where a user is unknown to the system or where it is necessary to model a user quickly but not necessarily with great accuracy.

An important subfield of user modeling research is that of plan recognition (Carberry, 1989). More specifically, the process of intended plan recognition tries to identify the intentions of a user, based on their utterances and on knowledge obtained throughout a dialogue, in order to provide helpful responses. Early efforts in plan recognition research focused on determining a single plan based on the user's utterances (Kautz and Allen, 1986). Later research (van Beek, Cohen and Schmidt, 1993) investigated the idea of using clarification dialogues when there are ambiguities in the system's view of the user's plan. By asking appropriate questions, a system can resolve these ambiguities in order to establish the user's true plan. This latter research is particularly relevant to the work in this thesis. In some sense, our topic of considering the idea of costs of interaction, and whether or not the perceived benefits of interaction outweigh the costs, can be viewed as an extension of the topic of determining when to engage in clarification dialogues.

2.2.1 Research Challenges for User Modeling

To summarize, there are many challenges in employing user modeling techniques in artificial intelligence systems. There are the initial difficulties of acquiring information in order to initialize the user model and of choosing an appropriate representation for the user modeling information. The system must also make use of the

user model to adapt its behaviour in an appropriate way for each user (e.g., (Ardissono and Goy, 2000). Finally, the user model must be updated continually, as the system continues to make new observations about a user's knowledge, preferences and goals (Kobsa and Wahlster, 1989). A system with a well-developed and well-exploited user model can tailor its behaviour to the specific user involved and can therefore perform collaborative tasks more effectively.

Of particular interest for the research described in this thesis will be the use of user models in determining interaction strategies between the system and the user. In particular, a representation of the user's knowledge and willingness to interact will be used to help a system to decide whether or not to initiate interaction at any given time.

2.3 Intelligent User Interfaces

Collagen (Rich and Sidner, 1998) is an application-independent toolkit for developing collaborative interface agents. The collaborative interface agents that can be constructed using this toolkit are meant to work alongside users, helping them with a particular task. For example, in an earlier paper, Rich and Sidner (1997b) describe the design of an agent that assists users in planning air travel itineraries. Because both the human user and the agent can use the application's programming interface (API) to affect the state of the program, the Collagen agents can be accurately described as mixed-initiative systems (Rich and Sidner, 1998).

One of the most interesting aspects of this research is that the agents always maintain a segmented interaction history (Rich and Sidner, 1997b), keeping track

of the various steps and substeps that have been achieved by the system-user team so far in the collaborative dialogue. The user can review this history at any time, to remind himself of what has been accomplished so far or to return to an earlier point in the dialogue in order to redo a particular step in the problem solving.

The discourse processing in Collagen is based on the theory of discourse of Grosz and Sidner (1986). There are three main components to this representation: the linguistic structure of the discourse (grouping utterances into discourse segments), the attentional state of the discourse (describing how the focus of attention of the dialogue participants is shifting), and the intentional structure of the discourse (a "recipe tree" representation showing which components of a shared plan have been completed (Pollack, 1990; Lochbaum and Sidner, 1990; Grosz and Kraus, 1996)). Acts within a dialogue are then classified by the discourse processing algorithm according to how they contribute to the discourse segment currently in focus.

Although the work in this thesis does not employ the same type of discourse model as the one used in Collagen – with a focus stack, natural language generation, and so on – it does involve modeling the state of the dialogue with the user. This is done by keeping a record of certain aspects of the dialogue thus far, including the number of times that the system has requested help from the user, the times at which those interactions took place, and the estimated cognitive effort that those interactions required of the user. Based on this information about previous instances of interrupting the user, the estimated cost associated with bothering the user in the current situation is calculated. This bother cost then plays a significant role in the system's decisions about whether or not to initiate communication with

the user.

2.4 Principled mixed-initiative design

Research on mixed-initiative systems has been growing in recent years and mixed-initiative techniques have been incorporated into systems in various application areas (Ferguson, Allen and Miller, 1996; Burstein and McDermott, 1996; Cox and Veloso, 1997; Carberry, 1997; Walker et al., 1997a; Horvitz, 1999; Kortenkamp et al., 1997). Most designers, however, have simply come up with their own domain-specific mixed-initiative solutions to their particular problem of interest. What is needed is a principled method for designing such systems: one that takes into account information about the task, the user and the dialogue simultaneously.

In Chapter 3, we provide such a principled solution: one that relies on a system's ability to model the user, the task and the dialogue all at the same time. Systems using our approach will be able to make rational decisions about interaction by analyzing the benefits and costs of communication with the user. User modeling research, introduced in Section 2.2, plays a key role in this decision process, as the system's beliefs about the user's domain knowledge and willingness to interact with the system are critical components of our design solution.

Chapter 3

A Model for Reasoning about Interaction

As discussed in Chapter 1, in this thesis we address the problem of designing mixed-initiative systems that can make reasonable decisions about when to solicit additional information from potentially helpful users. We propose that a system should ask for further input precisely when the perceived benefits of this interaction exceed the expected costs. In this chapter, a set of guidelines will be presented for how to design a system that can make exactly this type of decision. With these guidelines clearly defined, it would then be possible for designers of mixed-initiative systems to provide their systems with the ability to reason in a principled way about interaction with the user.

As an example of an application for which users have widely varying preferences in terms of the frequency of interaction, consider the design of *recommender systems*: programs that attempt to suggest items that users might enjoy, such as

movies, music, books or television shows (Kautz, 1998). These recommendations are usually based on some combination of explicit preference information that the user has provided and implicit observations of the user's tendencies. Different researchers have investigated the use of content-based recommendations (Pazzani, Muramatsu and Billsus, 1996) – analyzing how an item is similar to others that the user likes – and collaborative filtering (Konstan et al., 1997) – looking at how an item has been rated by users who are considered similar to this user.

One might expect that some users would prefer to have the system simply observe their habits and make recommendations accordingly, while other users would be interested in having a higher level of involvement. In fact, Buczak, Zimmerman and Kurapati (2002) have identified three classes of users in their work on television recommender systems. *Do it for me* users prefer a completely automated system that does not bother them at all, *Let's do it together* users are interested in having some moderate amount of control over the system's behaviour, while *Let me drive* users like to be totally in control of the system.

This categorization illustrates a need that will be emphasized throughout this thesis: a need to reason explicitly about the particular user involved before deciding whether or not to initiate communication. A user who prefers a completely autonomous system would likely be frustrated quickly by a system that asked frequent questions in an effort to improve the accuracy of its preference model. On the other hand, a user at the other end of the spectrum might be disappointed to learn that the system had decided to pass up an opportunity to involve the user. If systems are going to make rational decisions about interaction, then issues such as the

user's willingness to interact and the user's perceived knowledge about the domain must be taken into account. As a result, our proposed model for reasoning about interaction in mixed-initiative systems incorporates a user modeling component, focusing on the model of the user's knowledge and willingness to interact.

3.1 What to ask

Before presenting the details of our model, it is important to consider what types of questions a system would want to ask of a user. In this section, we identify several possible lines of inquiry that might come up during an intelligent system's reasoning process. The type of setting we are addressing is one in which the system has been enlisted by the user to perform some task. Although the system is able to make many decisions on its own, we are considering the situations that arise in which it might benefit from additional information or guidance from the user.

The focus of this thesis is on the system taking the initiative to ask questions of a user and on providing systems with a systematic way in which to make decisions about this type of initiative. Using Chu-Carroll's categorization of dialogue initiative versus task initiative (Chu-Carroll and Brown, 1997), our focus is clearly on the dialogue initiative interpretation. With respect to task initiative, it could be argued that a system taking the initiative to ask the user a question is in fact not *taking* the initiative, but *relinquishing* it by yielding control of the task to the user. This is in fact similar to the view adopted by Brainov and Hexmoor (2003), in which the *autonomy* of a system is defined as its ability to act efficiently without a user's help; this comparison is discussed further by Cohen and Fleming (2003).

It should be noted that, in designing mixed-initiative systems, it is also important to consider the fact that *users* will sometimes take control of the dialogue or the problem-solving, possibly at unexpected times. Systems should be designed to handle such events. Although we will discuss this aspect of the problem in Section 6.1.4, the focus of this thesis is on making decisions about system initiative: when should the system initiate interaction with the user?

Providing systems with a systematic way to make decisions about asking questions of the user, although it is only a piece of the overall puzzle of designing mixed-initiative systems, is a very important component. This section outlines some of the different motivations that a primarily autonomous system might have for interacting with a user.

- **Uncertainty about the environment.**

  In many situations, the system might lack information about the problem-solving environment or facts about the always-changing world, and it might believe that the user has more current knowledge. If this information is important and if the system is sufficiently uncertain (or completely ignorant), then it will be worthwhile to ask the user (if the latter is believed to be knowledgeable). Examples of questions of this type include:

  - (In the domain of transportation planning...)
    What is the current location of truck #517?
  - (In the domain of travel planning...)
    Is traffic currently heavy on route A?

- (In the domain of sports predictions...)

  Is Shaquille O'Neal expected to play tonight?

- **User preferences and goals.**

  Another important class of questions involves user preferences. In many task-oriented dialogues, it is the *user's* goals that the system is trying to achieve. Although many of the most important preferences can be expected to be provided during the initial specification of the problem, it is quite possible that certain preference information will be missing and will have to be obtained by the system at run-time. Of course, these questions should only be asked if it is believed that their answers will have an impact on the decisions that the system will make.

  A good example can be taken from the travel agent domain. A human travel agent usually will not ask a client a barrage of questions at the very beginning of a dialogue. Only the most important information (departure city, destination city, dates) will be gathered initially. More detailed preferences will only be requested if they are believed to be relevant.

  For example, there is no need to ask the user what city they would prefer for a stop-over if it might turn out that there are flights with no stops at all. As another example, it is probably safe to assume that a client will prefer a cheaper flight, all other things being equal. However, suppose the travel agent begins searching for flights and learns that the cheapest flight involves two stop-overs while there is also a (more expensive) direct flight. At this point, it would seem worthwhile to elicit the user's preferences about such

trade-offs, if the system has no reliable information already.

This class of questions differs from questions about the environment in that the user can almost always be expected to know about his own preferences.[1]

In fact, questions about user preferences might be the most common questions of all. They can be categorized into several different subclasses:

- *General preferences about one single variable.* For example, in the domain of travel planning,
  - Rank the airlines in order of preference.
  - Do you prefer Delta or United?
  - Indicate your restrictions on airline options.
- *Trade-offs among different factors.* These questions are closely related to the utility elicitation questions of Chajewska, Koller and Parr (2000).
  - Rank all possible combinations of factors. (Unlikely in practice.)
  - Indicate a preference between two choices. For example, in the travel domain, all other things being equal, would you prefer a $400 direct flight or a $300 flight with one stop?
- *Preferences about the dialogue itself.* For example, the model proposed in this chapter gives the user the option to modify a value that represents

[1] However, we should take care to account for the possibility that users might not *care* about certain aspects of the problem. If this is believed to be likely, then the expected value of asking a question should be quite low. Furthermore, although users might have a general sense of their preferences, it might be quite difficult for them if they are asked to quantify something that they are not used to thinking of in numerical terms.

the degree to which he is willing to interact with the system. This value, in turn, affects the frequency with which the system asks the user questions and gets him involved in the problem solving.

As with all potential questions for the user, the key observation is that some type of value-of-information analysis must be performed before asking a question. If the system *were* to ask a question, how would it change the decisions that would be made and how would it affect the expected success of these decisions?

- **User knowledge and abilities.**

  In some circumstances, it might be beneficial to ask the user to evaluate his own knowledge or abilities with respect to a particular aspect of the task. Although a user model might contain relevant information about users in general or user-specific information that has been deduced from observing the user in similar situations in the past, it might be quite useful to inquire about the user's own beliefs regarding his aptitudes. This could be beneficial not only for the current problem-solving sessions, but also as background knowledge for future interactions with this same user. Some examples of this line of questioning might include:

  - (In a math tutoring domain...)

    How would you rate your general knowledge of trigonometry, on a scale of 0 to 10?

  - (In the domain of providing the user with intelligent assistance with

computer programming...)

Do you consider yourself an expert, intermediate or novice programmer?

- **Domain-specific suggestions.**

  One of the main benefits of mixed-initiative interaction is that either the user or the system can take charge of directing the problem solving at any given time. If the system is involved in such a dialogue with a fully collaborative human partner, there are situations in which the system's best option is to yield decision-making power to the user entirely. This is particularly appropriate in situations in which human intuition about high-level domain goals has proven in the past to be superior to machine analysis. In such scenarios, the system might allow the user to take control and direct the system toward focusing its computational efforts on a certain subsection of the search space. For example, Anderson et al. (2000) discuss an example where the performance of a vehicle routing system can be improved when the user is allowed to take actions like moving a particular customer onto the delivery route of a particular truck.

Before concluding this section, it is important to point out that decisions about interaction are not limited to decisions about whether or not to ask questions of a user. Equally important in some domains is the question of whether or not to *provide the user* with additional information. Depending on its beliefs about the user's beliefs, the most beneficial action from the system's perspective could very well be to impart additional knowledge on the user, in order to bolster his knowledge about the task and to improve the potential results of future collaboration. Gmytrasiewicz

and Durfee (2001) discuss the analogue of this situation in a multi-agent setting. They examine the expected utility of sending a message to another agent to inform the other agent about some aspect of the world or about the "speaker" agent's own intentions.

An example of a scenario where it is important to *tell* a human user facts about the domain or about the system's current beliefs is the application of intelligent tutoring (Freedman, 1997; Aist, 1997; Lester et al., 1997; Shah and Evens, 1997). One of the constant challenges for a tutoring system is to make decisions about when it is best to let a student learn on his own, when it is best to provide subtle hints and when it is best to jump in and provide the student with information explicitly.

In other domains, it might also be beneficial to provide the user with information about the current state of the problem solving. If we are in a situation in which the user has not been involved in the task for some time, it might be worthwhile to communicate with the user solely for the purpose of establishing context and keeping the user informed of the present situation. In fact, Burstein and McDermott (1996) identified context registration as one of the key issues that needs to be addressed in the design of mixed-initiative planning systems.

One final note is that communication about the problem solving itself might be useful in certain contexts. For instance, in mixed-initiative settings, it could be important to ensure that the system and user have properly negotiated the roles of the two parties or that they have established a detailed plan for how the work will be divided during the session. The Collagen system (Rich and Sidner, 1998)

considers this by using a shared display for the user and agent, and by including a diagram of the history of the dialogue.

Although our focus in this thesis is on the value of asking questions in a system-user setting, it is important to emphasize that an interactive system in general must be concerned about the value of all aspects of communication with the user.

3.2 Factors in decision-making

3.2.1 General architecture

We will now describe the general architecture of our model, identifying factors that should play a role in a system's decision about whether or not to interact with a user. One of the main aims of the thesis is to provide a concrete characterization of these factors, to enable decision making about interaction.

To begin, consider the general situation in which an artificial intelligence system might find itself at any given time. It is common practice in artificial intelligence to speak of the system's current *state*. A state representation should capture everything that a system needs to know to distinguish its current situation from any other situation in which it might find itself while performing a task.

It will be proposed here that the current state of any interactive system should depend on a number of factors, each of which can be classified as belonging to one of three general components of the system: the task model, user model, and dialogue model. This is similar to the distinction made by Lambert and Carberry (1991) in their tripartite plan-based model of dialogue. They distinguish between domain

actions, problem-solving actions, and discourse or communicative actions. In our model, we combine domain and problem-solving actions into a single task model, and we add a user model as an additional component to the system.

1. **The task model.** This is a representation of the high-level tasks that the system might be attempting to accomplish in a problem-solving session. This model might include several "recipes" (Grosz and Sidner, 1990): specifications, for each task, of the subtasks that must be accomplished in order to complete that task. The task model should also include information about the current state of the problem-solving session - in a recipe, for instance, which subtask is currently being worked on and which ones have already been completed.

2. **The user model.** A user model is a system's internal representation of a user's knowledge, abilities, preferences, goals and any other user-specific information that helps a system to adapt its behaviour in a way that is appropriate for each user (Kobsa and Wahlster, 1989; Kass and Finin, 1988). This is a crucial component of any system that is designed to make rational decisions about interaction. In many cases, the decision of whether or not to interact will depend heavily on the specific user involved. For example, if the system is trying to decide whether to ask the user for a piece of factual information that is missing from its knowledge base, it should be much less likely to ask a user for help if it has evidence in its user model that this particular person is unlikely to have the required knowledge to answer the question. Most decision-making algorithms do not bother to model this, but simply assume

that a valuable answer will be returned, if the user is asked. We claim it is critical to model the user's knowledge, to get a more accurate evaluation of the utility of trying to ask for information, at this point in the processing.

3. **The dialogue model.** A model of the ongoing dialogue between the system and user is also a critical factor in decisions relating to interaction. With a record of the exchanges that have already taken place between the system and user during this session, the system has a view of what has already been conveyed to the user. Perhaps more importantly, a dialogue model allows the system to keep track of how frequently and how recently it has bothered the user during a problem-solving session. Different users might have different preferences about how involved they wish to be in the completion of a task. If, in a problem-solving session, a system has already asked several questions of a user who has indicated a generally low desire to be an active participant in the task, it should be more careful about interrupting this user again in the near future – perhaps initiating such an interaction only when the question is perceived to be particularly critical (Fleming, 1998).

   In this thesis, we will concentrate on modeling the general level of turn-taking, to measure user willingness. We will not elaborate further on methods to explicitly model dialogue structure and focus. In Section 6.1.1, we comment on the need to accommodate a richer model of dialogue, as part of the decision making, for future work. In particular, we discuss how modeling the *understandability* of an interaction for a user is an important factor to determine whether to interact. This factor will be influenced by the proper modeling of

the user's view of the current dialogue.

Overall architecture

Figure 3.1 shows the general architecture for this model. One of the key contributions of this thesis will be to specify exactly what elements of a task model, user model and dialogue model should contribute to decisions about interaction and how each of these factors should affect such decisions.

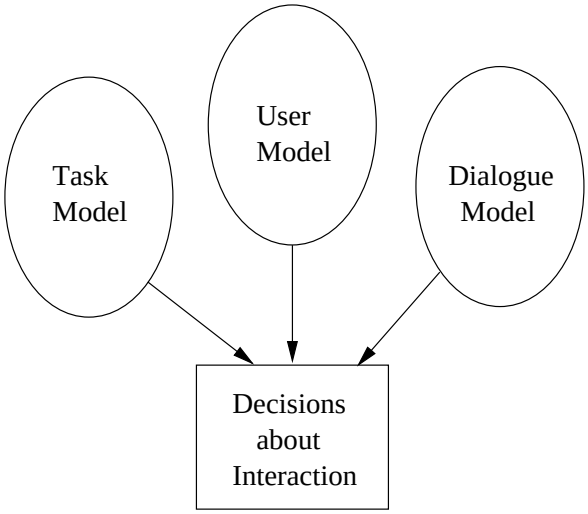


Figure 3.1: Influence of task, user and dialogue models

One way to view the system's reasoning about its current situation is by using the state-based representation that is common in artificial intelligence systems. In its current state s, there could be several actions that a problem-solving system might consider performing next. These include task-related actions that will help to move the problem solving closer to completion, as well as communicative actions

that are designed to garner more information from a user (or from some other source) in order to better equip the system to make decisions later in the session. In this thesis, the focus will be on communicative actions. In particular, we are interested in providing the system with a framework for determining the inherent value of asking a particular question at a specific time.

For the purposes of this thesis, we will assume that there is a module within the system that is responsible for generating potentially appropriate questions for the user, whether they be in the form of natural language utterances or menu selections or in some other form. Since this is not the focus of the research, the exact workings of this module will not be discussed, but it can be assumed that candidate questions would be generated on the basis of the current state of the system, what it intends to do next, and what information would be valuable to obtain.[2]

In many situations, there might be *several* potential questions at any given time. In such cases, the system's responsibility will be not only to determine *whether* or not to ask the user a question, but also *which* question appears to have the most value.

We also make some assumptions about the ability of the system to understand the intentions of the user. We are not focusing on resolving ambiguities involving goals and plans, or involving what the user actually said. Several researchers (e.g. (Paek and Horvitz, 1999a; Gmytrasiewicz and Durfee, 2001; McRoy and Ali, 1999)) have looked at the idea of possible miscommunications in dialogue; such

[2]If this module is not working well, or if there are errors in recognizing the plans of the user, there will be additional challenges to making the right decisions about interaction. However, these possibilities will not be considered in this thesis. This topic is discussed in Section 6.1.5.

methods would have to be incorporated into any practical mixed-initiative system making use of our model.

3.2.2 Specific factors

We will now elaborate on the general architecture presented in the previous subsection, listing a number of factors that should be considered by a system in attempting to make optimal decisions about interaction with a user. Although a few of these have been mentioned in passing already, it will be useful to establish a complete list of the relevant factors.

In this section, the factors will be identified and discussed at the qualitative level only. This is meant to provide the reader with an understanding of the intuition behind the inclusion of each factor. Later in this chapter, various approaches will be discussed for combining these factors into a quantitative decision-making framework for the system.

User model

In order to make rational decisions in an interactive setting, it is crucial to take into account the knowledge and preferences of the particular user involved in the dialogue.

Although there might be some additional domain-specific variables to consider, we have identified the following domain-independent user modeling factors:

- **The user's knowledge.** How likely is it that the user will have the required knowledge to answer the question? A system's beliefs about user knowledge

could span a wide range along each of two dimensions: the specificity of the user being modeled and of the subject matter being considered.

Along the first dimension, the system might have information about *this specific user*, about users who belong to the *same stereotype* as this user and about *all users* in general. Along the other dimension, it might have information about the likelihood of a user knowing about a specific concept, about concepts that are in a particular "class" or it might simply have a global value representing the probability of a user knowing any fact in this domain. Figure 3.2 illustrates the entire spectrum of knowledge about users' expertise.

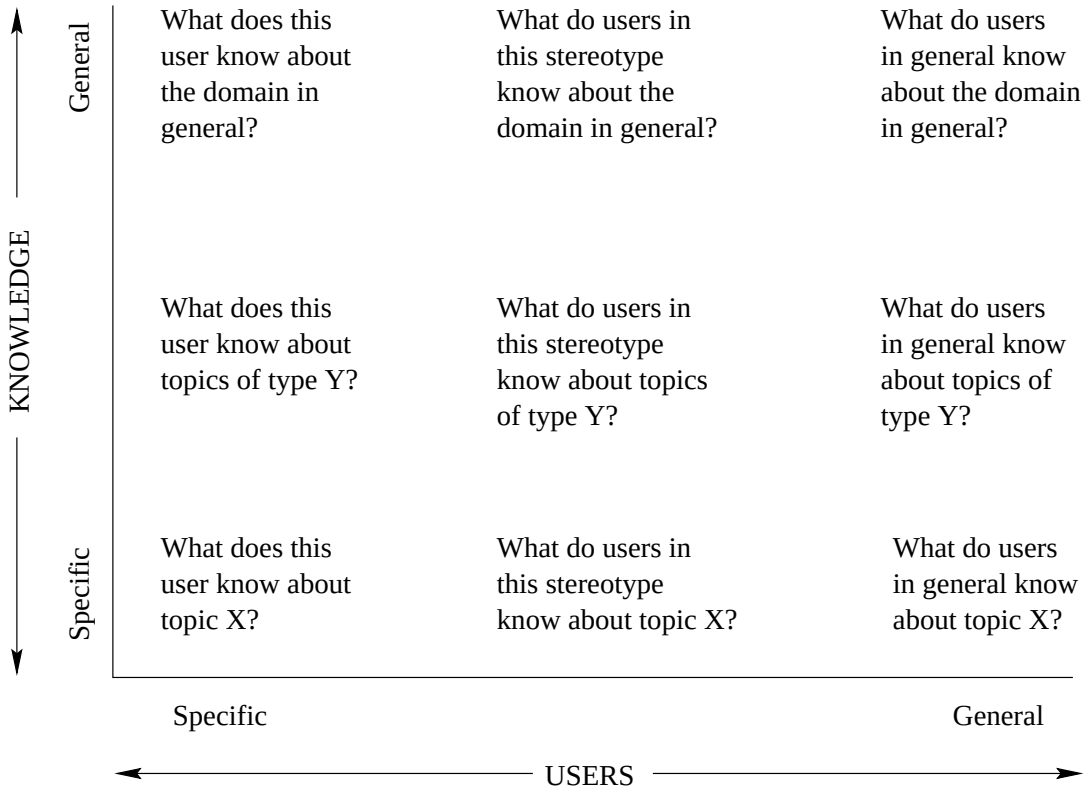


Figure 3.2: Spectrum of knowledge about user expertise

The manner in which these different types of user modeling information are balanced for a particular application domain will be up to the system designer. However, we provide in Section 3.8 a general model for how to make use of these multiple types of modeling information. The most important concept to mention at this point is that specific evidence should override general information. If a system knows (from previous interactions) that this user is very knowledgeable on a particular subject, then this information should take precedence over probabilistic evidence that users of this type are *unlikely* to know about the particular class of subjects under which the current question under consideration is categorized (Chin, 1989).

- **The user's willingness.** How has the user rated his willingness to be an active participant in this dialogue? All other things being equal, we should be less likely to ask a question of a user who has indicated a preference to be involved only in urgent situations than of a user who has requested a role as an active collaborator.

  This will often depend on the domain. In some cases (e.g., course advising (van Beek, Cohen and Schmidt, 1993; Cohen, Schmidt and van Beek, 1994)), any user should expect to be fully involved in the dialogue if they are to anticipate anything useful coming out of the session. The more interesting cases arise in domains where there might be a great deal of variation across users in terms of their willingness to interact with the system. An example

of such a domain is television personalization; see, for example, the work of Buczak, Zimmerman and Kurapati (2002).

The notion of user willingness must be considered in conjunction with the system's record of previous interruptions to the user. This idea will be discussed further in the upcoming section on the dialogue model.

- **The user's preferences.** This point is actually relevant both to this *User model* section and to the *Task model* section to follow. What do we know about the user's preferences regarding different outcomes in this domain?

  In many situations, these preferences themselves might be the topic of the proposed interaction. In the travel domain, asking a user if they prefer to fly with *Air Canada* or *WestJet* has very little usefulness if we have already learned from previous sessions that this user has a strong dislike for one airline or the other. However, this same question might be important to ask a different user, when the only airline preference information available is a summary of preferences from the general user population.

Dialogue model

A system must have an accurate record of how the dialogue has progressed so far. Of particular interest are the following two points:

- **Current context and understandability.** What information has been shared between the two parties up to this point? How aware is the user of the current state of the problem-solving? When the system is deciding whether or

not to ask the user a question, it must consider whether or not the user will have enough background contextual information to be able to understand the question and the system's underlying goals for asking it (Fleming and Cohen, 2000). In such a situation, the system must decide whether it is most useful (1) to ask the question in the hope that the resulting dialogue will go smoothly and that any necessary clarification dialogues will not be too costly, (2) to present the user with additional contextual information *before* proceeding with the interaction, or (3) to forgo interaction entirely, having determined that the likelihood of the user understanding is too low or that the expected cost of the interaction is too high.

- **Previous interruptions.** How many times has the user been bothered by the system so far in the dialogue? How recently have those interruptions taken place? Does the system have some estimate of the cost of the earlier interruptions, according to its perception of the cognitive effort that would have been required of the user to answer the questions and of what the user was busy doing at the time of the disruption? All of this must be considered along with the idea of user willingness, discussed above. The cost of bothering a user who has identified himself as a willing collaborator should be much lower than the cost of interrupting an unwilling user.

Task model

- **System uncertainty.** The task model is often what drives the system to consider interacting with the user in the first place. The system must consider

asking questions of the user only when it is not clear how it should proceed with the problem solving on its own. For example, if the system is provided at the outset with a detailed step-by-step procedure for how to solve a problem, and if it has all the information that it will need in order to complete each of the required steps, then there should be no need to communicate with the user. The exceptions to this are: (1) communication in order to maintain the user's awareness of how the problem solving is progressing, and (2) communication initiated *by the user* to provide additional information to the system or to inquire about the status of the session.

- **Task criticality.** What do we know about the criticality of this task? Are we necessarily looking for the *best* solution? Is it acceptable simply to find a "good" solution (one that meets certain minimal criteria)? Is it our goal to find *any* solution to the given problem? Phrased differently, what is the expected tolerance for suboptimality in this domain? How serious are the consequences if the problem is not solved perfectly?

- **Utilities of states.** In addition to the information about the possible high-level tasks or goals and how they should be achieved, the task model must contain information about the system's perception of the desirability of being in each of the possible states in which it might find itself over the course of a problem-solving session. The idea of utility theory is discussed in Section 3.3.

- **Time.** A key factor in reasoning about the value of interacting with a user is an estimate of the amount of extra time that the interaction would take. Included in this issue is the question of how time-critical the task is. Even if an interaction with the user might take a significant amount of time, this might not be a concern if the task is one for which there is no deadline or no clear advantage to completing it quickly. However, if a task must be completed in a timely manner, interaction will be advantageous only if it is expected to improve the quality of the solution significantly and still allow the task to be completed on time (Xuan, Lesser and Zilberstein, 2001).

- **Other costs.** In most domains, there will other costs that must be modeled. For example, if certain domain actions consume resources, such as CPU time, disk space, or access charges to commercial resources, then these costs must be incorporated into the task model as well.

3.2.3 Summary

Table 3.1 provides a summary of the main factors identified in this section. The factors included in this list will play a central role in the rest of this chapter, as we develop a domain-independent model for reasoning about interaction with users. The second column classifies each factor as being relevant to the user model, dialogue model or task model.

Factor	Model
The user's knowledge	UM
The user's willingness to interact	UM
The user's preferences/utility function	UM, TM
Task criticality	TM, UM
Current context and expected understandability of system utterance	DM, UM
Previous interactions	DM
The expected improvement of the system's task performance due to interaction	TM
Time and time criticality	TM
Resource costs and other task-specific costs	TM

Table 3.1: Summary of factors to be used in model

3.3 Details of the Model

3.3.1 Utility theory

Our model will involve using the idea of expected utility to guide the system in its decisions. A utility function U is a function that assigns a value to every state in the state space for a problem.[3] This value is meant to represent the usefulness or desirability of being in that state. If a decision-maker prefers to be in state s_A over state s_B, then $U(s_A)$ should be greater than $U(s_B)$.

When forced to decide from among multiple actions at a given time, the decision-maker can consider the new states that would result from performing each of the possible actions and then choose the action that will lead to the most desirable state.

An agent in an *uncertain* environment will often face situations in which it is

[3]The state space is the set of all states that might possibly arise in the process of solving the problem.

unsure of the exact effects of its actions. For example, in a state s_0, taking action A might lead to any of several states s_1, s_2, $\cdots$, s_n with different probabilities. The *expected utility* of an action can be calculated by summing the utilities of all possible outcomes of the action, weighted by the probabilities of those outcomes.

More formally, let E represent the evidence that the agent has gathered about the world and let $Result_i(A)$ represent the possible outcomes of performing an action A. Then, the expected utility of action A given evidence E is computed as follows (Russell and Norvig, 1995):

$$EU(A|E) = \sum_i P(Result_i(A)|E, Do(A)) \times U(Result_i(A))$$

"The principle of **maximum expected utility** (MEU) says that a rational agent should choose an action that maximizes the agent's expected utility." (Russell and Norvig, 1995)

For example, suppose that a decision-maker has the choice of two possible tasks to perform. If the first one is chosen, there is a guaranteed reward of \$100. If the second task is selected, there is a 20% chance of being rewarded with \$1000 and an 80% chance of no reward at all. Which task should be chosen?

In the first case, the value of the reward is known to be \$100. With the second option, the *expected* reward is (0.20)(\$1000) + (0.80)(\$0) = \$200. We will assume for the sake of this simple example that the utility depends linearly on the amount of money earned, although research (Grayson, 1960) has shown that this is not generally the case for human decision-makers. If we do make this assumption, then the second task above is the better choice because of its higher expected utility.

3.3.2 Single-Decision Problems

Having discussed in Section 3.2 the intuition behind our model for reasoning about interaction with users, we will now introduce the actual details of the model. We will begin by examining the special case of a system working on a task during which there will be only *one* opportunity to make a decision about interaction. The general scenario faced by such systems is illustrated in Figure 3.3. The system first makes a decision about interaction with the user. Whether or not it chooses to interact, it then makes a decision about which action to perform and then takes that action to complete the task.

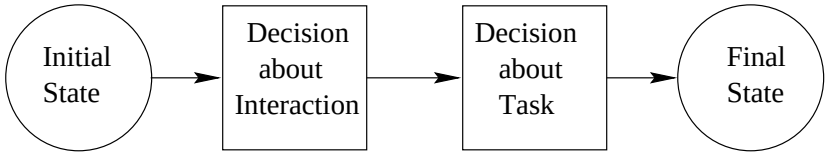


Figure 3.3: A single-decision problem

As will be discussed in the next subsection, such a simple system will be rare in practice. However, starting with a simple example will help to illustrate the basic ideas behind our model and to motivate later discussion of more complex decision problems.

Our general approach to reasoning about interaction is as follows. Given a question that the system is considering asking the user, perform the following steps.

1. Determine the expected benefits of interacting with the user. More specif-

ically, determine by how much the system's performance on the task is expected to improve (if at all) after asking the user the question.[4]

2. Determine the expected costs of the interaction.

3. Proceed with the interaction only if the benefits exceed the costs.

Benefits

The following formula is used for computing the *benefits* of interaction. Let EU_{ask} represent the expected utility of the outcome(s) that would result if it *did* ask the user the question and then chose an action based in part on the user's response. Let $EU_{\neg ask}$ represent the expected utility of the outcome(s) that would result from the system making its decision without any additional interaction at this point. Then, the benefits are computed simply by taking the difference between these two values.

$$\text{Benefits} = EU_{ask} - EU_{\neg ask}$$

When we mention utility in this context, we are referring to the expected value of the problem solution that is reached in some final state. This does not take into account any costs that might be involved in reaching the solution. As will be seen soon, such costs are treated separately in our model.

EU_{ask} and $EU_{\neg ask}$ themselves are computed by summing over the possible outcomes in each case, weighted by the probability of each outcome.

[4] This is essentially the expected *value of information* (Howard, 1966) associated with asking the question.

Let us begin with $EU_{\neg ask}$, the expected utility of the action that the system would take without any further interaction. For each possible non-communicative action in the set *Actions* that the system is able to perform, it computes the expected utility of that action by summing over the utilities of all possible outcomes of the action, weighted by the probabilities of each of those outcomes. The rational action for the system to choose is then the action with the highest expected utility; $EU_{\neg ask}$ is equal to the expected utility of this best possible action. Again, in the formula below, $Result_i(A)$ represents the possible outcomes of performing the action A. Note that we have simplified the notation from the expected utility formula in Section 3.3.1. In the equation below, $P(Result_i(A))$ replaces $P(Result_i(A)|E, Do(A))$, the probability that $Result_i(A)$ would occur given that the agent has gathered evidence E about the world and has performed action A. This simplification is made to improve the readability of this formula and others to follow in this section.

$$\begin{aligned} EU_{\neg ask} &= \max_{A \in Actions} EU(A) \\ &= \max_{A \in Actions} (\textstyle\sum_i P(Result_i(A)) \times U(Result_i(A))) \end{aligned}$$

One issue that has been largely ignored in earlier work on utility-based reasoning is the fact that attempts to gather more information, particularly when dealing with human collaborators, are not guaranteed to be successful. To account for this, our calculation for EU_{ask}, the expected utility of the outcome that will result if we *do* interact with the user includes a new variable. Let P_{UK} represent the probability

that the user will actually have the knowledge required to answer the question.[5] Also, let *Resp* represent the set of possible responses to the question.[6] Then EU_{ask} is computed as:

$$\begin{aligned} EU_{ask} &= P_{UK}[EU(\text{if user knows})] + (1 - P_{UK})[EU(\text{if user does not know})] \\ &= P_{UK} \sum_{r \in Resp} P(\text{Resp} = r) EU(\text{S's action choice} | \text{Resp} = r) \\ &\quad + (1 - P_{UK}) EU_{\neg ask} \end{aligned}$$

In words, EU_{ask} is the sum of two terms. The second of these terms captures the case in which the user is unable to answer the question. In this case (occurring with probability $1 - P_{UK}$), the system would simply fall back on what it would have done without interacting (with expected utility $EU_{\neg ask}$). The first term considers the case in which the user *is* able to answer the question (probability P_{UK}). The expected value of the system's actions in this case is found by considering all the possible responses that the user might give and the expected utility of the action that the system would take in each of those cases.

In the formula above, $EU(\text{S's action choice} | \text{Resp} = r)$ is calculated by taking the maximum of the expected utilities of all possible actions:[7]

$$\max_{A \in Actions} \left(\sum_i P(Result_i(A)) \times U(Result_i(A)) \right)$$

[5] As will be seen in Section 3.8, this is in fact a *function* that depends on the user and on the questions being asked; however, we will treat it here as a single variable for the sake of readability.

[6] Section 6.1.6 will include a discussion of what can be done in situations where the system might not know all the possible responses to every question.

[7] Again, the notation in this formula has been simplified, with $P(Result_i(A))$ replacing $P(Result_i(A) | E, Do(A))$. The evidence E would now include the system's observation of the user's response to the question.

Considering Incorrect User Responses

In some cases, the expected utility of the system's choice of action after receiving a response r from the user will depend on the system's prior beliefs about whether or not the user's response will in fact be correct. For example, it is possible that the system might decide to take a particular action A if it receives response r from the user. However, if the user's response were actually incorrect, then A might not turn out to be a very sensible action at all. If the system expects that there is a significant risk of the user providing a misleading answer, it should incorporate this into its reasoning.

This possibility can simply be weaved into $EU(A|\text{Resp} = r)$, where A is the system's choice of action given that the user provided r as a response. For instance, if there were two other possible responses, r' and r'', for the question, then the expected utility of the system's choice of action would be

$$\begin{aligned} &P(\text{Correct} = r|\text{Resp} = r) \times U(A|\text{Correct} = r) \\ +\ &P(\text{Correct} = r'|\text{Resp} = r) \times U(A|\text{Correct} = r') \\ +\ &P(\text{Correct} = r''|\text{Resp} = r) \times U(A|\text{Correct} = r'') \end{aligned}$$

In the general case,

$$EU(A|\text{Resp} = r) = \sum_{r' \in Resp} P(\text{Correct} = r'|\text{Resp} = r) \times U(A|\text{Correct} = r').$$

The formula above relies on the fact that the system would be able to estimate the probability of every possible error the user might make: for example, if the user says the answer to a multiple-choice question is A, what is the probability that the correct answer is actually B, C, D or E? In most practical applications, it

is unlikely that a system designer would be able to provide the system with such detailed information. However, what would be realistic and useful to model is a single value representing the probability that the user's answer to a question will actually be correct. If this value is known, then the other possible answers can simply be given a probability based on the system's prior beliefs. For example, suppose the system initially had no information and believed that all answers were equally likely. If the user gives a reply of A and if the system believes that the user is correct 90% of the time, then the probabilities of B, C, D and E being the actual correct answer would each be 2.5%.

In many domains, it might be reasonable to assume that users will *always* be correct (or that the system should not take it upon itself to question the user's authority). However, the possibility exists to take this factor into account if a system designer should choose to consider it.[8]

Adding understandability

In addition to the possibility that a user might not have the knowledge required to answer the question, it is possible that a user might not be able to provide immediate help to the system because he does not understand precisely what is being asked. This might be due to a failure to understand the system's terminology or due to the fact that the user simply hasn't been involved enough in the ongoing problem solving to be able to fully appreciate what the system is asking. In Section 6.1.1, we

[8]Note that checking for possible *inconsistencies* between one response from a user and previous responses from that same user may also lead the system to conclude that the user is incorrect. This would require some consistency checking machinery.

will discuss how understandability could be incorporated into our model in future work.

Costs

The *costs* of interaction in our model are represented using a linear model: the total cost is a weighted sum of any individual cost measures C_i that have been identified for the application domain.[9] Each of these factors is normalized so that the possible values range from 0 to 100, with a cost of 0 indicating no cost at all and a cost of 100 representing the maximum possible cost in this application domain.

$$\text{Costs} = \sum_i w_i C_i$$

Note that each of these cost measures C_i is actually a cost function that might depend on the current state, on the particular action or question being considered, and/or on certain components of the user model.

In fact, as we did with benefits, we should be careful to represent not just how much an interaction costs, but how much *more* it would cost than doing things without asking the user. The actual formula for costs is

$$Costs = Costs_{ask} - Costs_{\neg ask}$$

[9]We believe that, in general, the cost measures will be mutually preferentially independent, and so an additive cost function is appropriate. (Russell and Norvig (1995) define mutually preferentially independent as: "Two attributes X_1 and X_2 are preferentially independent of a third attribute X_3 if the preference between outcomes $< x_1, x_2, x_3 >$ and $< x_1', x_2', x_3 >$ does not depend on the particular attribute x_3 for attribute X_3.")

where $Costs_{ask}$ represents the additional costs that will be incurred if the system asks the user a question, and where $Costs_{\neg ask}$ measures any additional costs that will occur if the system does not ask the user. Each of the two terms is computed as a weighted sum of the various component costs, as shown above.

However, to simplify our discussion throughout the thesis, we will combine these two sums into one weighted sum of terms C_i, to measure the *additional* cost associated with asking the user. Any cost measures that will occur only if the user is asked will carry positive weights in the equation, while costs that are incurred only if the user is *not* asked will carry negative weights. For example, if the only alternative to asking the user for help in a particular situation is to send a query to a remote database, there will be some cost associated with this retrieval. Such a cost would be represented as a cost of *not* asking the user for help and would be incorporated into our sum with a negative weight attached to it.

In the subsequent example, the cost of interaction will be measured by a weighted sum of only two cost factors: t, the cost associated with the estimated additional time required for the interaction, and b, the cost associated with bothering the user in the current situation. These are both costs associated with asking the user for help and will have positive weights in our formula. We will discuss other potential cost measures and the specific details of how the time and bother costs are computed in Section 3.4.3.

A single-decision example

We will now present a simple example to illustrate the application of our model to decision-making about interaction.

In this example, the system has been asked to plan a travel route for the user to get from City A to City B. It determines that it has the choice between two different paths. According to the system's knowledge, path 1 is shorter but is congested (due to heavy traffic) about 50% of the time. Path 2 is significantly longer, but it is never busy.

Benefits of interaction

In this example, the system has access to a utility function for this type of situation. The utility function assigns a value to each possible outcome in the domain, where an outcome consists of the decision that was made by the system (which route did it choose to take?) and the actual state of the world (was path 1 in fact congested?). For example, one outcome would be that the system opted to take path 1, but found that it turned out to be busy and, therefore, slow.

The values assigned by the utility function in this example are meant to capture the attitudes of the average user toward different possible outcomes in this domain, and are shown in Table 3.2.[10]

The ideal outcome in this example would be if we were to choose path 1 and if it were to turn out to be clear. The worst outcome would involve choosing path 1 and then finding out that it is congested. In between these two extremes, choosing path 2 is a fairly safe decision, but we would be somewhat less pleased if we were

[10] The range used for utility values varies across the literature, but it is fairly common to use a scale of 0 to 1 or 0 to 100. The latter will be used in our examples. It is also common to assign the minimum and maximum values in the range to the least and most desirable outcomes, respectively, and to use these initial assignments to guide the assessment of intermediate utility values (Keeney and Raiffa, 1976). The actual scale used is irrelevant as long as the chosen values are consistent with the preferences being represented.

to choose path 2 when, in fact, the shorter path had been available.

System's choice	Actual state of path 1	Utility
Path 1	Path 1 clear	100
Path 1	Path 1 busy	0
Path 2	Path 1 clear	50
Path 2	Path 1 busy	70

Table 3.2: Utility function for path example

Now, suppose that we have reason to believe that the user might have access to recent traffic information and could therefore help with the decision-making. We believe that there is a 60% chance that the user has accurate traffic information.

To make the problem slightly more interesting, we will assume that there is an additional 10% chance that the user has traffic information that turns out to be *incorrect*.

The remaining 30% is assigned to the case in which the user states that he has no additional traffic information. In this case, the system should fall back on what it would have done without asking the user.

Let us first consider the decision that the system would make in the absence of any further information from the user. In other words, we want to compute the value of $EU_{\neg ask}$, the expected utility of the best action the system could take if it did not ask the user the question.

There are two possible actions in this simple example: choose path 1 or choose path 2. According to the problem description above, there is a 50% chance that

path 1 will be congested. The expected utility of choosing path 1 is computed as follows:

$$\begin{aligned} EU(A = \text{path 1}) &= \textstyle\sum_i P(Result_i(A)) \times U(Result_i(A)) \\ &= P(\text{path 1 clear}) \times U(A = \text{path 1} \mid \text{path 1 clear}) \\ &\quad + P(\text{path 1 busy}) \times U(A = \text{path 1} \mid \text{path 1 busy}) \\ &= 0.5\ (100) + 0.5\ (0) \\ &= 50 \end{aligned}$$

Similarly, the expected utility of path 2 depends on the system's current beliefs about the state of path 1.

$$\begin{aligned} EU(A = \text{path 2}) &= P(\text{path 1 clear}) \times U(A = \text{path 2} \mid \text{path 1 clear}) \\ &\quad + P(\text{path 1 busy}) \times U(A = \text{path 2} \mid \text{path 1 busy}) \\ &= 0.5\ (50) + 0.5\ (70) \\ &= 60 \end{aligned}$$

$$EU_{\neg ask} = \max_{A \in Actions} EU(A) = \max(50, 60) = 60.$$

Therefore, with no additional information, it appears that the best solution for the system is to play it safe and choose path 2, since the expected utility of path 2 is higher than the expected utility of path 1.

Now that we know the expected utility of what the system could do *on its own*, let us consider the possible outcomes if the system *were* to ask the user for further information. Table 3.3 summarizes the possible scenarios that might arise.

User response	Actual state of path 1	Probability	System's choice	Utility
Path 1 clear	Path 1 clear	0.30	Path 1	100
Path 1 clear	Path 1 busy	0.05	Path 1	0
Path 1 busy	Path 1 clear	0.05	Path 2	50
Path 1 busy	Path 1 busy	0.30	Path 2	70
No answer	Path 1 clear	0.15	Path 2	50
No answer	Path 1 busy	0.15	Path 2	70
Overall expected utility of system choice after asking				71.5

Table 3.3: Possible scenarios in path-choosing example

The probabilities in Table 3.3 are computed by considering both the system's beliefs about whether or not the user will have the knowledge and the system's prior beliefs about the actual state of the path. For example, the system believes that the user will know the correct answer with a probability of 0.6. Since the system initially believed that path 1 was equally likely to be clear or congested, there is a probability of 0.3 of the user correctly saying that the path is clear (Row 1) and a probability of 0.3 of the user correctly saying that it is busy (Row 4).

The overall expected utility (71.5) shown in the final row of Table 3.3 is computed by summing over all the possible outcomes, weighted by their probabilities. This tells us that, in the average case, interacting with the user will lead the system to a choice with an expected utility of 71.5. Recall that, without asking the user, the expected utility of the system's best action – simply choosing path 2 as a safe route – was 60.

Our conclusion is that, despite the fact that the user might not know the answer or might even *mislead* the system, there is a clear expected benefit to requesting the additional information from the user. We represent this benefit by looking at the

expected gain in performance on the task if we were to interact – in other words, by taking the difference between the expected utility after interaction and the expected utility of the system's default action in the absence of further interaction.

In this example,

$$\text{Benefits} = EU_{ask} - EU_{\neg ask} = 71.5 - 60 = 11.5$$

Costs of interaction

It is important now to consider the fact that there are also *costs* involved in interacting with the user.

As presented earlier, costs are calculated with a weighted sum over all cost measures C_i that have been identified for the domain.

$$\text{Costs} = \sum_i w_i C_i$$

In this example, the cost of interaction will be measured by a weighted sum of *two* cost factors: t, the cost associated with the extra time required for the interaction, and b, the cost associated with bothering the user in the current situation. We will discuss other potential cost measures in Section 3.4.3.

Also in Section 3.4.3, we will discuss in more detail how a system would determine the actual values for each of the cost measures and for the weights attached to each factor. However, to keep this example straightforward, we will simply state that the time cost is 10 on a scale of 0 to 100 (the interaction will not take long at all), the bother cost is 10 (the communication will not be perceived by the user as

being very bothersome), and the weights associated with time and bother are 0.2 and 0.3, respectively.

With these values, the total cost associated with the interaction is:

$$\begin{aligned}\text{Costs} &= w_t t + w_b b \\ &= 0.2(10) + 0.3(10) \\ &= 5\end{aligned}$$

Since the benefits of interaction were computed earlier to be 11.5, the benefits outweigh the costs and our system's optimal decision would be to proceed with the interaction with the user.

Table 3.4 shows some variations on the above example, demonstrating how the system's decisions about interaction are affected by modifying the values of the relevant factors. For example, while the first row summarizes the earlier example, the second row shows that if the system had believed that the user was not very likely at all to be able to answer the question (with all other factors remaining unchanged), the ultimate decision would have been instead to forgo interaction. The third and fourth rows show that if the scenario had been changed so that the system is initially almost certain that Path 1 is clear or almost certain that it is congested, then interacting would not be beneficial at all.[11] The fifth row shows a case where, despite the fact that the system was initially quite sure about the state of path 1, it still decides to interact because it is certain that the user will know

[11] The *negative* value for benefits in fact demonstrates that the system would likely *decrease* its performance on the task if it were to ask the user, since the likelihood of obtaining new information is quite low and since there is a chance of actually being misled by the user's response.

the answer. The final two rows demonstrate the effects of modifying the values of the two cost factors.

The graphs in Figure 3.4 provide a different way of looking at two of these variations. In part (a), the probability of the user knowing the answer is varied while all other factors are kept constant at the values shown in row 1 of Table 3.4. The benefits exceed the costs only when P_{UK} is greater than 0.34. In part (b), the system's belief about the probability of path 1 being clear is varied while all other factors are kept constant. In this case, the benefits exceed the costs only when this probability is roughly between 0.26 and 0.68.

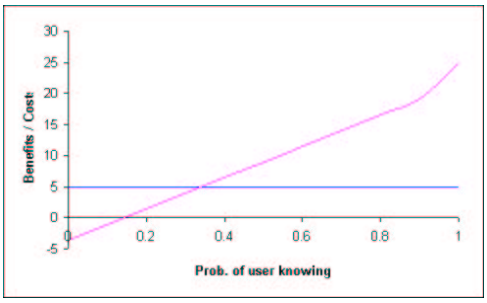
The utilities of the possible outcomes are assumed to be the same in all cases.

P(U knows)	**P(Path 1 clear)**	t	b	EU_{ask}	$EU_{\neg ask}$	**Benefits**	**Costs**	**Ask?**
0.6	0.5	10	10	71.5	60	11.5	5	Yes
0.2	0.5	10	10	61.5	60	1.5	5	No
0.6	**0.9**	10	10	89.7	90	-0.3	5	No
0.6	**0.1**	10	10	64.7	68	-3.3	5	No
1.0	**0.9**	10	10	97	90	7	5	Yes
0.6	0.5	**50**	10	71.5	60	11.5	13	No
0.6	0.5	10	**40**	71.5	60	11.5	14	No

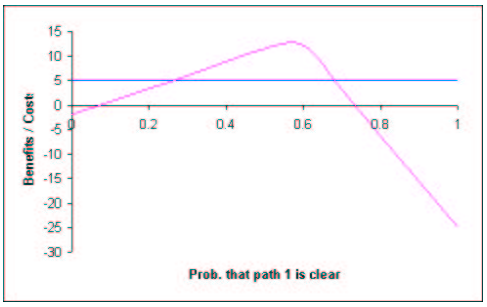
Table 3.4: Variations on path-choosing example

3.3.3 Sequential Decision Problems

Examples as simple as the one presented in the previous section are unlikely to arise in practice. One of the reasons for this is that this example involved a system that was trying to perform a task involving only *one* decision. Once this decision was



(a) Varying the probability of the user knowing



(b) Varying the probability of path 1 being clear

Figure 3.4: Variations on the path-choosing example

made, the task was complete and the system was able to observe the consequences immediately. In practical problems, a system will normally be faced with many decisions, one after another. The reward associated with performing an action might not be immediate. The full implications of a decision made at some point early in the process might not be observed until the entire task has been completed. In order to determine the true value of an action, we would have to project very far into the future to see what might happen along the way, and what solutions and rewards might be reached. This is referred to as a sequential decision problem (Russell and Norvig, 1995).

If computation time were not an issue, and if the task were guaranteed to finish after a finite number of steps, then the model in Section 3.3.2 could be applied directly to the sequential decision case by using a dynamic programming algorithm (Bellman, 1957). If utilities are known for final states (at a final time t), then these could be used to calculate the utilities of states at time $t-1$, and so on until a utility value is attached to every possible state in the state space.

However, if a system is to be making decisions about interaction at run-time, this type of reasoning is computationally infeasible for anything but the smallest of problems.

One potential solution to this is to try approaching the problem of reasoning about interaction as a Markov decision process (MDP) (Puterman, 1994). This formalism is a natural extension of the ideas presented in this section, and is discussed in Section 3.5. It allows a *policy* to be generated before the system runs, specifying the optimal action for every possible state that might arise.

Before discussing the MDP approach, however, it is important to investigate in more detail how the benefits and costs from this section are computed, and how these calculations would be extended to the case of a sequential decision problem.

3.4 Benefits and Costs

In this section, we provide some further discussion about the benefits and costs of interacting with users.

3.4.1 Benefits

We begin this section by repeating the formula for benefits in the model described in Section 3.3. Let EU_{ask} represent the expected utility of the outcome(s) that would result from the system choosing an action if it did ask the user the question. Let $EU_{\neg ask}$ represent the expected utility of the outcome(s) that would result from the system making its decision without any further interaction. Then,

$$\text{Benefits} = EU_{ask} - EU_{\neg ask}$$

EU_{ask} and $EU_{\neg ask}$ themselves are computed by summing over the possible outcomes in each case, weighted by the probability of each outcome. As mentioned earlier, one of the main concerns in trying to model the benefits of interaction is in determining utilities for the different possible outcomes that might arise. In anything but the simplest possible domains, it might be difficult to elicit utility values for every possible state in the state space, even if it is possible to devise an additive utility function that can be used to evaluate a state on the basis of a number of subutility functions.

In Section 3.6, we address the issue of how to estimate the value of asking a question heuristically, without knowing a full utility function. Prior to that section, however, we will assume that the system has access to a utility function that assigns a value to every state in the state space, or one that assigns a reward to every *final* state and can be used to compute expected utilities for non-final states. This function is meant to represent how good a particular solution is, irrespective of any costs involved in reaching this solution.

3.4.2 Possible Extensions

In addition to reflecting expected improvements in the system's problem solving for the current session, the designer of a system might also choose for the value of a solution to include long-term benefits: expected changes in the system's knowledge or user model that are likely to help with *future* sessions that might occur. For

example, if the system is likely to be working with this user on similar problems in the future, then an outcome in which the system has solved the problem but has also learned a great deal about the user's preferences should be considered more valuable than accomplishing the same task without learning for the future. This idea is related to the work of Sullivan, Grosz and Kraus (2000), in which agents in a multi-agent system take into account both the immediate reward they can expect from performing an action and the expected future income associated with the action.

On a related note, a system should not only consider the effect of an interaction on its *own* knowledge, but also on the knowledge that a user possesses. There should be significant benefits to an interaction that will result in the user having improved expertise on a topic, either if this knowledge is likely to come into play in future problem-solving sessions or simply if it is believed that the user will value having had the opportunity to learn from the experience. This is particularly relevant in domains such as intelligent tutoring, where one of the primary goals of the system is indeed to help the user to acquire new knowledge or abilities.

Throughout this thesis, it is assumed that the system designer will ensure that such concerns are incorporated into the utility values that are assigned to different states in the state space for the particular application domain.

Another relevant concern is that of incorporating future likely interaction load into the decision of whether or not to ask a question (Shifroni and Shanon, 1992; Raskutti and Zukerman, 1997). If asking or not asking a question at the present time will result in the need to ask other questions at a later time, then this should

be considered when computing the value of asking the question currently being considered. Similarly, if the system expects to require the user's assistance for very crucial matters at a later time, then it might be best to forgo interaction on a less critical topic now. If a system is, in fact, fully examining all possible future actions and interactions, then this will be incorporated. The difficulties associated with doing a complete analysis of all future events are discussed in more detail in Section 3.5.

3.4.3 Costs

Again, the *costs* of interaction in our model are represented using a linear model: the total cost is a weighted sum of any individual cost measures C_i that have been identified for the application domain. Each of these factors is normalized so that the possible values range from 0 to 100, with a cost of 0 indicating no cost at all and a cost of 100 representing the maximum possible cost in this application domain.

$$\text{Costs} = \sum_i w_i C_i$$

Two major cost factors that will apply in any application domain are time and bother. Although costs related to time have been modeled in decision-theoretic systems (Horvitz and Rutledge, 1991), the factor of bother has been modeled by relatively few researchers (Bauer et al., 2000; Raskutti and Zukerman, 1994; Godden, 2000; Fleming, 1998). The "bother factor" in this thesis is meant to represent the degree to which a user would be annoyed, disrupted or inconvenienced by any

interaction with the system.[12]

The exact role of this particular factor will depend on the domain. For some domains, users will almost always enter the collaboration with the understanding that they will be expected to play a major role in the problem solving. In other systems, which will be of greater interest from the perspective of our model, the possibility exists for a wide range of participation levels for users. This latter case arises in applications that are not necessarily meant to be interactive, but where the system has been charged with the responsibility to perform some task and might occasionally benefit from obtaining more information from the user at certain points in its reasoning. This is related to work on adjustable autonomy (Hexmoor, Falcone and Castelfranchi, 2003), in which systems can adapt the degree of autonomy they exhibit in different situations. The connection between our work and research in adjustable autonomy will be discussed further in Section 6.2.5.

More importantly, the role of the bother factor will depend on the individual user. While some users will prefer to be very actively involved with these systems, doing everything in their power to help the system achieve the best possible results, others will prefer not to be bothered and will be happy with the best solution the system is able to find on its own.

It should be pointed out, however, that even systems that are meant to be completely interactive must take care not to bother a user unnecessarily. Consider again the idea of a system that is intended to help a user with planning a vacation.

[12]Note that Raskutti and Zukerman (1994) model the annoyance due to asking irrelevant questions, ones that do not pertain to the user's actual goal. In this thesis, we are assuming that a separate module is determining which question to ask. Only if this module is functioning improperly will irrelevant questions arise. This topic is discussed further in Section 6.1.5.

The number of questions that *might* be useful for a system to ask is quite large. However, if the system were to ask every one of these questions at the beginning of a planning session, even the most patient user would very quickly become overwhelmed and frustrated. Instead, it seems reasonable for a system to ask only those questions that are absolutely necessary at the outset. From that point on, decisions should be made about the value of asking a question. The expected cost of bothering the particular user involved should be an important factor in such decisions.

The specific details of how to represent time and bother costs, as well as other potential costs of interaction, are presented in the upcoming subsections.

Time Cost

Two primary methods have been used for incorporating time into utility-based decision-making in the past (Horvitz and Rutledge, 1991).

In the first, there is simply a constant penalty for each time step that is executed. The utility of performing action a in state s at time step t is $U(a, s, t) = U(a, s, t_0) - ct$, where t_0 is time 0 and where c is some constant penalty for each time step that elapses.

In the second method, there is an exponential decrease in the value of performing an action as time elapses: $U(a, s, t) = U(a, s, t_0)\beta^t$, where $0 < \beta \leq 1$.

Although the precise *form* of the time-dependent utility function should be up to the system designer, the first approach will be used in the examples discussed in this thesis.

The system designer might opt to give the user an important role in specifying the time cost: that of providing the parameter c or β. In some circumstances, only the user could estimate what the tradeoff is between time and solution quality. The value of this constant can be interpreted as a measure of the *time criticality* of the problem. If time is critical, then the constant c should be high (or β should be low) – the cost of delaying action is higher for more time-critical situations. If time is completely irrelevant, as long as a solution is obtained eventually, then c could be set to 0 in the first approach or β to 1 in the second.

An additional possibility is that of a deadline. In some applications, a solution might have no value at all if it is not found before some cut-off time T. This is fairly simple to incorporate. If the task must be completed by time T, then anything done after time T has utility 0.

Bother Cost

A very important component of our model is the consideration of the user's *willingness* to collaborate with the system and the frustration that might result from bothering the user excessively.

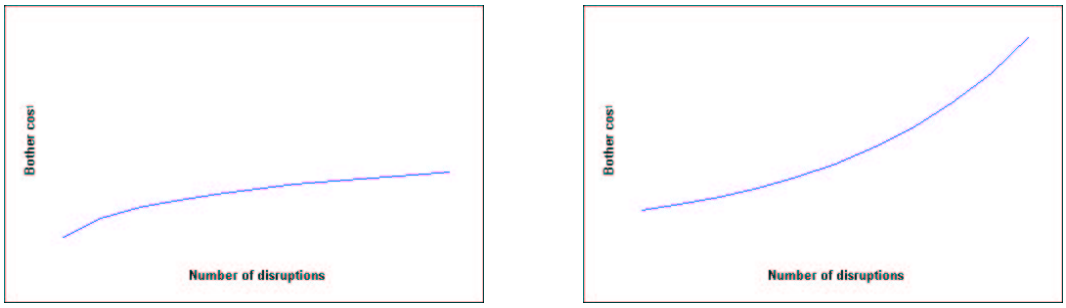
To make things as easy as possible for the user, the idea of willingness to interact should be set up on a simple scale such as 0 to 10. We describe later in this section how this willingness value is used to estimate the cost of bothering the user. Any future adjustments to the willingness value would be made by the user according to intuition. If he feels like he is being bothered too much, then he should move the level down to indicate that he is less willing to interact than he had indicated earlier.

Since it is often difficult for humans to view such subjective matters on a numerical scale, some guidelines should always be available to the user to help with the initial selection of an appropriate willingness value. In fact, we recommend that the system ask the user a series of questions when the system is first employed. The responses to those questions could be used to show the user where he falls on that scale of 0 to 10. This could be done by asking questions similar to the utility elicitation questions of Chajewska, Koller and Parr (2000). The user could be presented with a series of hypothetical situations, where he is asked to assume that every possible outcome in the problem domain can be rated on a scale of 0 to 100. The questions would then take the form of "If the system could achieve a performance of x without asking you any questions or a performance of y by asking you n questions, which alternative would you prefer?" After asking a small number of these questions, with different values of x, y and n, the system could use the responses to find the best fit between the user's attitude toward disruptions and the bother functions we will develop later in this section.

Again, the actual *cost* associated with bothering the user with a particular question will then depend on the user-defined willingness factor. In the single-decision example of Section 3.3.2, this willingness factor is the only concern when it comes to estimating the cost of bothering the user. However, in a sequential decision problem, the bother cost should also incorporate a measure of how much the user has been bothered in the dialogue so far.

In their work on programming by demonstration, Bauer et al. (2000) used a scheme in which the shape of an annoyance function is determined by the esti-

mated level of patience that a user has: a very patient user would have a log-like penalty function for disruptions, while an impatient user would have an exponential function.[13] This is illustrated in Figure 3.5.



(a) Patient user (b) Impatient user

Figure 3.5: Typical shapes of bother functions

We adopt a similar approach of having the *shape* of the annoyance function determined by the user's self-reported willingness level, but extend it by determining where we are on the x-axis of that annoyance function at a given point in the dialogue not solely by the number of disruptions so far, but also on information about when those interruptions took place and on an estimate of how bothersome each interaction was. In essence, recent interruptions and difficult questions should carry more weight than interruptions in the distant past and very straightforward questions. Further research effort must be devoted to determining an appropriate

[13]This information is based on personal correspondence in 2002 with M. Bauer (German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany). No published record of the exact method has been found.

function for the cost of bothering the user. However, according to the desired behaviour just described, we propose the following function for the bother cost.

For every past interaction I with the user, let $t(I)$ be the amount of time that has elapsed since that interaction. The specific implementation assumed in this thesis involves dividing time into discrete time steps and using the number of steps as the value for $t(I)$.[14] Let $c(I)$ be an estimate of how bothersome the interaction actually was, in terms of the cognitive effort required of the user to answer the questions. Then, we use the following formula estimating the "bother so far" (BSF).

$$BSF = \sum_{I} c(I)\beta^{t(I)}$$

to give us an idea of how bothersome the dialogue has been so far. The term β is a discount factor, $0 < \beta \leq 1$, that is used to accomplish the goal of diminishing the impact of interactions that took place a long time ago.

Suppose we bothered the user 2 time steps ago, 7 time steps ago and 13 time steps ago. Assuming that each interaction had a cost[15] of $c(I) = 1$ and that $\beta = 0.95$, our estimate of "bother so far" is $1 + 0.95^2 + 0.95^7 + 0.95^{13} = 3.11$.

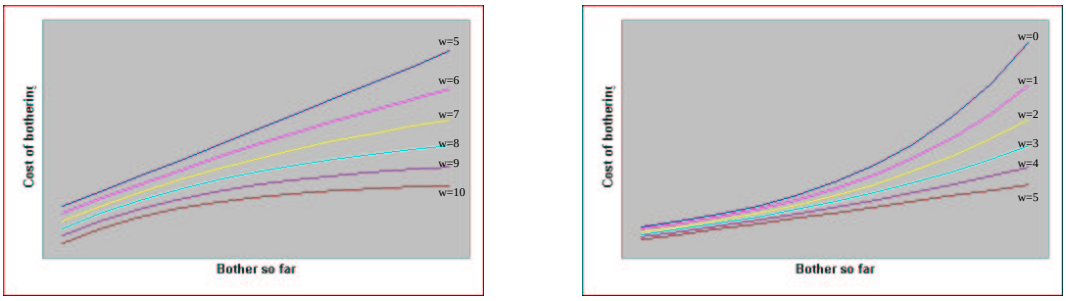
The willingness of the user to interact (the variable w, on a scale of 0 to 10, introduced earlier) is then incorporated as follows. We define two new terms $\alpha = 1.26 - 0.05w$ and $Init = 10 - w$. [16] The bother cost is then computed as $bother =$

[14] In domains where time is better treated as a continuous variable, the system designer can adjust the formula so that $t(I)$ is the actual time elapsed divided by some pre-determined constant.

[15] In this example, we assume that all interactions are equally costly; for some domains, system designers might choose to make certain types of questions more or less costly.

[16] These are simply suggested values; the system designer might consider doing some empirical research to determine the most appropriate values for a given domain. The proposed formula for

$Init + \frac{1-\alpha^{BSF}}{1-\alpha}$. Figure 3.6 shows the graphs for the bother functions of willing and unwilling users.[17]



(a) Willing users (b) Unwilling users

Figure 3.6: Bother functions for willing and unwilling users

Suppose, for example, that the user is a very willing one, with $w = 9$. Then α would be $1.26 - 0.05w = 0.81$. If the bother so far is computed to be 3.11 as shown above, then the cost of bothering would be

---

α is intended to give a nearly linear bother curve for users with moderate willingness values and bother curves with more exponential and logarithmic appearances, respectively, for more unwilling and willing users (see Figure 3.6). The value of $Init$ is intended to reflect the cost of bothering a user for the first time. Our choice for $Init$ assumes that this cost will be negligible for a very willing user ($w = 10$) and quite high for an unwilling user ($w = 0$).

[17] The intuition for this formula is as follows: If BSF happens to be an integer $(0, 1, 2, 3, \cdots)$, then the formula for bother will work out to be $Init, Init + 1, Init + 1 + \alpha, Init + 1 + \alpha + \alpha^2$, *etc.*. The more the user has been bothered in the past, the most bothersome this potential interaction would be. If α is greater than 1 (willingness $w \leq 5$), then each additional increment to BSF results in more bother cost than the previous increment; this is done to yield an exponential bother function for unwilling users. If α is less than 1 (willingness $w > 5$), then each additional increment to BSF results in less bother cost than the previous increment, yielding the more logarithmic graph shape for willing users. The function is continuous so that non-integral values for BSF can be handled as well.

$bother = \frac{1-\alpha^{BSF}}{1-\alpha} = \frac{1-0.81^{3.11}}{1-0.81} = 2.53$

If we had bothered the user more frequently and more recently, say at 1,4,6,9 and 15 time steps ago, then the bother so far would be 4.73 and the cost of bothering would be 3.32.

For a less willing user (one with $w = 1$), the bother costs for the same two situations described above would work out to be 3.85 and 6.97, respectively.

Task criticality

We also recommend that the criticality of the task be modeled as a potential cost of *not* interacting. As stated earlier, utilities of different outcomes are often assigned so that the best possible outcome has a utility of 100, while the worst possible outcome has a utility of 0. However, this fails to account for the fact that the worst possible outcome in some domains might be far more disastrous than the worst possible outcome in other domains.

Suppose, for example, that a system has been set up to order lunch automatically for employees in an office. If the system makes an error and orders an undesirable meal for a particular person, then this outcome would be quite unsatisfactory for that employee and would be assigned a utility of 0 for the task. However, the negative impact of this decision pales in comparison to the consequences that might be observed in a military domain, for example, where failure to gather sufficient information might cause a system's poor decision to lead to the loss of human lives.

Let $Taskcrit$ represent the criticality of the task, on a scale of 0 to 100. Again, this variable would be assigned a default value by the system designer, but individual users would be free to adjust this value in most application domains. Suppose

the system were to take an action leading to an unsatisfactory solution, and that this outcome could have been avoided if the user had been asked for help. We view this possibility as a potential cost of *not* interacting with the user. As discussed in Section 3.3.2, such a cost is incorporated into the cost formula with a negative weight.

This concept of task criticality is closely related to the idea of tolerance for suboptimality. If a domain is one where users will only be satisfied with *optimal* behaviour on the part of the system, then the task criticality should be set to be extremely high. Subsequently, the system will be more likely to ask for user assistance before making decisions on such a critical task. On the other hand, the task criticality can be set to a low default value for domains in which users are willing to accept any solution that satisfies a given set of constraints.

Resource Costs

Another important aspect to the consideration of costs of interaction is the consumption of system resources. For example, in some domains, the system (1) might have to perform several database queries (which might come at some cost), (2) might have to communicate with *other* agents if information cannot be obtained from a user (bother cost analogous to the user bother, *plus* cost of communication channel, etc.), (3) might have to use CPU time, memory, disk space and other computational resources.

These costs will be domain-specific, but any that are deemed to be relevant should be incorporated as a factor C_i in the costs formula introduced earlier:

$$\text{Costs} = \sum_i w_i C_i$$

This is especially important when comparing the value of asking the user a question and the value of performing other actions to try to obtain this information. Interaction with the user might take some time and might inconvenience the user. However, if the information involved is essential for the system, and if the only alternative to asking the user is an extremely expensive query to a remote database, then the interaction with the user should turn out to be the better choice.

Other costs

In some domains, system designers might identify certain task-specific costs that should be modeled. For instance, in the work of Walker et al. (1997a), it is important for the spoken dialogue agents to minimize dialogue costs such as the number of automatic speech recognition errors and the number of help requests from the user. Such costs can be incorporated into the weighted sum formula presented at the beginning of this section.

3.4.4 Determining Weights

The cost formula in the model includes weights on the various component cost factors. The intention is for these weights to be assigned initially by the system designer, most likely in consultation with a domain expert. These initial weights should reflect the relative contribution that the designer believes is appropriate for each of the component costs with a typical user. However, in most or all domains,

it should be possible for these weights to be adjusted by the system as it learns more about the domain and about the user – or by the user himself.[18]

In this section, we present some ideas on how the initial weight assignments should be determined and on how to allow weight adjustments once the system is deployed.

Initial Weight Assignments

The cost values C_i are intended to be normalized on a scale of 0 to 100, with a cost of 0 indicating no cost at all with respect to that particular factor and a cost of 100 being the maximum possible cost. Since all costs are then on the same scale, the weights w_i should simply reflect the relative importance of each of the cost factors. If the cost of bothering the user is considered to be twice as important as the cost associated with elapsed time, then the weight attached to the bother cost should be twice as high.

Care must be taken as well to ensure that the weights are assigned so that benefits and costs can be reasonably compared. If, for example, the weights on the costs were all set to be too high, the system would seldom decide to interact since the computed costs would almost always exceed the expected benefits.

The amount of effort that is invested into choosing appropriate values for these weights is up to the system designer. Some designers might be comfortable simply establishing values for the weights based on their own intuition about the im-

[18] A possible exception to this is in highly critical domains in which system designers do not believe that the user should be provided with much autonomy in controlling the system's behaviour.

portance of each cost factor. Other designers might prefer to determine appropriate weights experimentally. For this, we would recommend an approach very similar to that used in the PARADISE framework for evaluating spoken dialogue agents (Walker et al., 1997b). In this framework, it is proposed that a system's performance on a task can be measured by correlating it with user satisfaction. After using a system, users are asked a series of questions (shown in Figure 3.7) to determine their overall satisfaction with the experience. By using these user satisfaction values as an indicator of overall task performance, the PARADISE framework then determines the weights on task success and on various dialogue-based cost measures by using linear regression. The learned linear function is then used as an evaluation function, approximating the expected user satisfaction, in later sessions. More detail about this work can be found in Section 7.2.

Weight Adjustments

For most applications, it will be reasonable to allow users to adjust the weights on the cost factors whenever they desire. For instance, a user who notices that the system is asking too many time-consuming questions in situations where time is somewhat pressing might consider increasing the weight associated with the time cost. A system should include a straightforward interface through which the user can inspect and adjust the weights on the cost factors.

Another possibility is to have the system learn from its experience and from user feedback when it makes incorrect choices. The system can then make small adjustments to the weights to improve its future decision-making. Since this is not the focus of this thesis, we will not discuss it any further here. We will as-

The following questions comprised the user satisfaction survey presented to users after interacting with the Elvis e-mail agent (Walker et al., 1997a):

- Was Elvis easy to understand in this conversation?
- In this conversation, did Elvis understand what you said?
- In this conversation, was it easy to find the message you wanted?
- Was the pace of interaction with Elvis appropriate in this conversation?
- In this conversation, did you know what you could say at each point of the dialog?
- How often was Elvis sluggish and slow to reply to you in this conversation?
- Did Elvis work the way you expected him to in this conversation?
- In this conversation, how did Elvis's voice interface compare to the touch-tone interface to voice mail?
- From your current experience with using Elvis to get your email, do you think you'd use Elvis regularly to access your mail when you are away from your desk?

Figure 3.7: User satisfaction questions from PARADISE

sume throughout the thesis that reasonable weights have been chosen and that any necessary adjustments will be made as needed.

3.5 Dialogues as Markov Decision Processes

We view the process of a system and user working together on solving a problem as a dialogue between the two parties. It has been suggested (Levin, Pieraccini and Eckert, 1998; Litman et al., 2000) that dialogues can be represented as Markov

decision processes (MDPs). In this section, we will investigate the MDP formalism as an extension of the model presented in Section 3.3.

3.5.1 Definitions

The Markov decision process (MDP) formalism is often very useful for intelligent agents that are faced with sequential decision problems. An MDP is the "problem of calculating an optimal policy in an accessible, stochastic environment with a known transition model" (Russell and Norvig, 1995). A policy is a mapping from states to actions: a complete specification of the best action to take in every possible state that might arise. Boutilier (1999) provides the following formal definition of a Markov decision process:

> A fully observable MDP $M = < S, A, Pr, R >$ comprises the following components. S is a finite set of *states* of the system being controlled. The agent has a finite set of *actions* A with which to influence the system state. Dynamics are given by $Pr : S \times A \times S \rightarrow [0, 1]$; here $Pr(s_i, a, s_j)$ denotes the probability that action a, when executed at state s_i induces a transition to s_j. $R : S \rightarrow \mathbb{R}$ is a real-valued, bounded *reward function.*

Other sources (e.g., (Xuan, Lesser and Zilberstein, 2001)) define the reward function as operating on $S \times A$ instead of on S only; that is, there is a reward associated with *performing an action* in a given state instead of for simply being in a particular state.

Once an agent's decision-making situation has been represented in the MDP framework, there are straightforward algorithms (e.g., *value iteration* (Puterman,

1994)) for determining the optimal policy for that agent. Such a policy specifies, for each state in the agent's state space, the best action to perform in that state. This is computed by determining which action will have the highest expected long-term reward in each possible state. Once the policy has been computed, the agent only needs to know its current state in order to look up the best possible action quickly.

3.5.2 MDPs for Dialogue: an example

In limited domains, the MDP formalism does indeed seem appropriate for dialogues. For example, Levin, Pieraccini and Eckert (1998) discuss an example where the goal of the system is simply to acquire the correct day and month from the user, while minimizing the length of the interaction. There are 411 states in their system, representing the different possible values that the system might have for the month and day (including the empty value). For example, one possible state is the one in which the system knows that it is March, but does not know the day. They consider four possible actions: asking the user for the day, asking for the month, asking an open-ended question for the date, and closing the dialogue. In their paper, they demonstrate how reinforcement learning can be used to learn an optimal dialogue strategy for this problem and for a small air travel information system.

In this section, we will present an example illustrating how such MDPs should be set up in general for problems involving human-computer dialogue, emphasizing the role of the factors that we have identified earlier in this chapter. In particular, we propose that factors such as the probability of the user being able to answer a question and the estimated cost of bothering the user with such an interaction

should be key factors in setting up an MDP for this type of problem.

As an example, consider the following toy problem, a modification on the Wumpus world problem presented by Russell and Norvig (1995), with the addition of the possibility of interaction with a user.

The idea is that we have an agent moving around in a grid, trying to get to a reward (gold) while simultaneously attempting to avoid running into a creature known as the Wumpus. In our version, we know the initial location of the Wumpus and we know that, at each time step, it stays where it is or moves in any of the four possible directions with equal probability. (If any of the four directions would involve hitting a wall, it stays where it is instead of moving in that direction.) Our agent, meanwhile, can choose to move in any direction, to stay where it is *or* to ask the user for the Wumpus's current location. If the agent does ask, there is some chance that the user will not know the answer.

The agent's current state is represented as a tuple $< A, T, W, E, N, D >$. A represents the agent's current location (always known), T the current time step, W the Wumpus's location when we last knew it for certain, E the number of time steps elapsed since we last knew W's location, N the number of times the user has been asked so far, and D whether or not the agent is dead.[19]

The "world" in this example is a 3-by-3 grid. Positions are labelled:

[19]We make an assumption that if we do run into the Wumpus, we will be alerted immediately and the task will end in failure. The reason D has to be included in the state representation is for cases like this: suppose the first five elements of the current state are <0,3,2,2,0>. We are in position 0 at time 2, the Wumpus *was* in position 2, but we have not asked in 2 time steps. If this were all that we represented, we could be alive or dead in this state, depending on what the Wumpus had done since we last asked. We have to include D to distinguish between these two possibilities – to determine whether or not the agent can still continue.

6	7	8
3	4	5
0	1	2

One possible initial arrangement would have the agent at position 0, the gold at 8 and the Wumpus at 7.

The agent's possible actions are: moving left, moving right, moving up, moving down, staying still, and asking the user for the Wumpus's current location.

In Section 3.3.3, we discussed the fact that one challenge to extending our single-decision model to the case of sequential decision problems was that it is difficult to evaluate the expected utility of every state without projecting into the future to consider all possible sequences of events. In the MDP framework, this work is done entirely by the value iteration algorithm (Puterman, 1994). Once the system designer has specified the state space, the set of possible actions, the reward function and the transition model, the value iteration algorithm generates an optimal policy mapping every state to the best possible action in that state.

In this modified Wumpus World example, the reward for getting to the gold is $R = 1000 - \beta t$, where t is the time when the gold is reached and β is a time discount factor which can be varied. We impose a time limit (in this example, $t = 7$) after which the agent must stop and therefore receives no reward.

We also impose a cost associated with bothering the user by asking for the Wumpus's location. In this example, we assume a constant cost for each instance of bothering.[20] We will discuss below how choosing different values for this constant

[20] The possibility of non-linear bother functions was discussed in Section 3.4.3.

affects the system's behaviour.

Now, the overall value of a final state s is specified by the reward $R(s)$ associated with that state. The time and bother costs are represented as negative rewards when performing actions involving interaction with the user. Once the state space, set of actions, transition model and reward functions are established, the MDP can be solved using the value iteration algorithm (Puterman, 1994). This will provide values for all non-final states in the state space. Once the MDP has been solved, we will have an optimal policy for the agent: a specification of the best action for each state that might arise.

Consider, for example, the state <4,4,8,1,0,0>: the agent is in square 4 at time 4, the Wumpus was in square 8 one time step ago, the user has not yet been bothered, and the agent is not dead.

In Table 3.5, we can see how the optimal action in the policy changes if we vary the cost of bothering the user and the probability of the user being able to answer the system's question. The expected long-term utilities of actions are computed by running the value iteration algorithm, and the time cost is set at a constant penalty of 30 for each time step.

Bother cost	**Prob. user knows**	**EXPECTED UTILITY**						**Optimal action**
		Ask	**Up**	**Down**	**Left**	**Right**	**Stay**	
10	0.8	326	322	0	0	322	318	Ask
0	0.8	336	322	0	0	322	318	Ask
20	0.8	316	322	0	0	322	318	Up/Right
10	**1.0**	330	322	0	0	322	318	Ask
10	**0.6**	321	322	0	0	322	318	Up/Right

Table 3.5: Optimal actions in the modified Wumpus World

These examples demonstrate that things behave intuitively. As the bother cost is increased, the value of asking decreases to the point that it is overtaken as the optimal action. Similarly, as the user becomes less knowledgeable, the expected value of asking decreases.

Note that there is an apparent difference between the decision procedure being used in the MDP model and that being used in the earlier model of Section 3.3.2. In the MDP model, instead of looking at whether the benefits of interaction exceed the costs, we are simply concerned with choosing the *best* possible action, whether that action involves interaction or not.

However, these two interpretations are in fact equivalent. In the model presented in Section 3.3.2, the benefits represented the degree by which the system's performance on the task was expected to improve if it did interact with the user. Specifically, benefits were computed by taking the difference between EU_{ask} and $EU_{\neg ask}$, respectively the expected utility of the outcome(s) that would result from the system choosing an action if it *did* ask the user the question and the expected utility of the outcome(s) that would result from the system making its decision *without* any further interaction.

The costs represented the degree to which interaction introduced *extra* cost to those costs that would be incurred anyway without interaction. As discussed in Section 3.3.2, costs are essentially computed as $Costs_{ask} - Costs_{\neg ask}$. So, if the benefits of interacting exceed the costs, then $EU_{ask} - EU_{\neg ask} > Costs_{ask} - Costs_{\neg ask}$. This can be rearranged as $EU_{ask} - Costs_{ask} > EU_{\neg ask} - Costs_{\neg ask}$.

Now, we can define the overall value of an action as the difference between its expected benefits and costs. With this approach, the final inequality above indicates that the value of asking exceeds the value of not asking. Since the value of not asking is in fact the value of the *best* action the system could take without asking, we can conclude that $Benefits > Costs$ implies that asking is in fact the best action. The reverse implication can be argued in a similar way.

As a further illustration of the effect of modifying either the probability of the user knowing the answer to a question or the cost of interaction, consider Table 3.6. This table shows the number of states (out of 9856 in the modified Wumpus problem) for which asking is the action with the highest expected utility. P_{UK} represents the probability that the user will have the knowledge required to help. Again, the results are intuitive. As the cost of communication increases and as the user becomes less knowledgeable, we can see from the table that it makes less sense to interact.

P_{UK} **Cost**	**1.0**	**0.8**	**0.6**	**0.4**	**0.2**	**0.0**
0	239	217	217	204	158	0
10	49	29	15	1	0	0
20	6	0	0	0	0	0
30	0	0	0	0	0	0

Table 3.6: Asking as the optimal action in the modified Wumpus World

The above discussion demonstrates that the MDP formalism *can* be used to model a situation where asking is one of the possible actions. By using expected

utilities, we can decide what is the best action for any possible state we might end up in. This leads to a policy that can be used by the agent to look up the best action just by knowing its current state.

The problem is that most domains are not as simple as this one. Very often, we might not be able to describe a state as easily, we might not be able to assign utility values to every possible situation that might come up, or we might not be able to predict the possible outcomes of each action (especially when the action involves asking the user). Also, the questions that we might ask the user might not be as straightforward and there might be ambiguity in the user's responses.

The MDP framework is in some sense an ideal model. *If* we had all of these things available, then we could reason in a rational way using theoretical techniques that have been well-studied and proven to work. However, in domains that are more complex, we will need to look at approximating the MDP approach.

3.5.3 MDPs for Dialogue: shortcomings

We maintain that, in a more complex, more practical dialogue situation, optimal decisions will often not only depend on the current state of the problem-solving, but also will rely heavily on the identity of the *user* that is involved in the session. While it might be ideal to ask user A a particular question when we find ourselves in state s, user B might find this same question intrusive, inappropriate or impossible to answer in the same context.

In order for an optimal policy in the MDP framework to accommodate this, the agent's state representation must include information from the user model and

from the dialogue history. Being in a particular problem-solving situation with user A, who has a particular set of knowledge and preferences and with whom the agent has a particular dialogue history, must be represented as an entirely different state than being in that same situation with user B, who has a different profile and dialogue history. In the previous example, this problem was avoided because P_{UK} was simply treated as a constant for the entire domain and because the bother factor was considered the same for any question at any time in the dialogue. In fact, these factors should depend on the current state and on the particular question that the system is considering asking the user.

With all the different factors that must be tracked for each user (willingness to interact, probability of knowledge, etc.), this is going to make the difficult problem of explosive state spaces in Markov decision processes even more serious.

Furthermore, a user model will rarely be a complete description of the user's current state of knowledge, preferences, etc. Rather, it will contain estimates of various characteristics of the user, based on observations that the system has made. This will often take the form of probabilistic information about the user.

One way to characterize the statements in the preceding paragraph is to say that such an agent is working in an *inaccessible* environment: the current state will not be known exactly at any given time. *Partially observable* MDPs (POMDPs) are designed to deal with this type of scenario. In POMDPs, decisions must be made when the agent does not have complete knowledge of its current state. Rather, decisions must be based on the observations the agent has made about the world (and its perceived *likelihood* of being in each of the possible states) (White, 1993). Un-

fortunately, exact algorithms for POMDPs are computationally intractable (Zhou and Hansen, 2001).

In addition to the issues of partial observability and computational intractability, we have identified three significant problems with the use of Markov processes as models of dialogue:

1. Throughout a dialogue, the task model and user model will change frequently as the system learns more about the domain and about the user's knowledge, abilities, preferences and goals. Since MDP methods involve the computation of an optimal policy *ahead of time*, such a policy would have to be recomputed regularly in order to properly reflect changes in the system's domain knowledge and user model. For anything but the simplest domains, recomputing the policy would be a computationally expensive exercise.

2. MDPs rely on all variables in the domain being discrete. In practice, many relevant variables will be naturally represented as continuous. Although values for such variables could always be broken up into discrete ranges, it would seem more appropriate to reason with the real-valued data itself.

3. Also related to the question of computing an optimal policy, we would suggest that a large majority of "possible states" in the representation of a particular domain will come up very rarely, if ever, in practice. The process of computing an optimal policy – a mapping from every state in the domain to its optimal action – wastes a great deal of time since much of the computed information will not be useful in practice.

3.5.4 Approximating MDPs

Exact solving of MDPs is rarely done anymore because it is far too expensive for practical problems (Schuurmans and Patrascu, 2001). A substantial amount of work has been published on methods for approximating Markov decision processes. These methods deal primarily with the problems concerning the explosive state space and the computational cost of the algorithms.

Dynamic decision networks (DDNs), for example, have been described (Russell and Norvig, 1995) as approximations to partially observable MDPs. Systems using a DDN representation are better able to deal with the fact that decision processes in dialogue are often made on the basis of partial information. They do not use an explicit state space like the one seen in MDPs; instead, a set of state variables is used to represent the current state. At any given time, a system might know values for some of these variables, but might only have probabilistic information about the values of other variables. Decisions can be made based on whatever partial information is available at the time of the decision.

Horvitz and Paek (2001) and Murray and Van Lehn (2000) demonstrate how dynamic decision networks can be used for specific tasks, such as modeling the uncertainty in a user's goals and providing intelligent tutoring.

Other approaches to approximating MDPs include the work of Schuurmans and Patrascu (2001). In this research, the nearest linear fit is found for the optimal value function for an MDP. Linear programming techniques are then used to solve the approximate problem. This approach has provided excellent speed improvements over earlier approximation methods, but has resulted in about twice the

approximation error.

However, despite advances in research on compact representations for MDPs, some of the problems cited earlier still remain, such as the need for discrete variables and the constantly changing user model.

A system that is able to make decisions "on the fly" by reasoning with the most up-to-date information, without having to recompute an optimal policy, seems more appropriate. Our model in Section 3.6 presents formulas for the system to use at run-time to make decisions about interaction according to its current information about several relevant factors.

3.6 An information-theoretic approach

In many practical application areas, it will be infeasible for a system to project ahead to all possible sequences of events that might arise in the future and to consider the probabilities and utilities of the possible outcomes.

In this section, we present an alternative approach to quantifying the benefits of an interaction. Instead of performing detailed reasoning about the decisions that the system might make, the possible consequences of those decisions and the probabilities of each of those outcomes, we propose a model that makes use of current information that any system might have available, without having to project into the future.[21]

[21] Please note that MDP researchers might object to the fact that, in using this approximation, we lose the Markov property because decisions are no longer based only on the current state of the system. We acknowledge this shortcoming and intend to consider it in future work.

3.6.1 Quantifying benefits

As discussed earlier, it will be possible in some applications to represent any situation that might arise as being one of a finite set of states, to determine probabilities of actions leading to transitions among those states, and to establish rewards for all outcomes that might arise. In such cases, the benefits of an interaction can be determined simply by computing the expected gain in performance that would result from the interaction.

This was the case in the path-choosing example in Section 3.3.2. Without asking the user for help, the system believed that Path 2 looked like a slightly better option. However, because there was a significant amount of uncertainty in the system's beliefs, the expected utility of choosing this path was only 60 on a scale of 0 to 100. By soliciting additional information from the user, the system knew it could then change its decision according to the response it received from the user, choosing the most appropriate path in either case. This would result in an expected utility of 71.5, representing a significant improvement.

In the absence of complete information about utilities and probabilities, we must resort to some sort of heuristic estimate. In this section, we consider a model that attaches a value to the actual *process* of asking a particular question. We begin with two major factors that should influence such an estimate: (1) the system's uncertainty about its current knowledge and (2) the perceived importance of the information.

The intuition behind the first point is that the system should be more likely to ask a question if it has no idea which of several possible facts is true in the

real world than, for example, if it believes that one of the facts is almost certainly true. With the second point, the idea is that a system should be far more likely to ask questions that have been identified as important than it should be to ask less crucial questions.

Uncertainty

What is needed is some means by which the system's uncertainty can be quantified. Let us first consider the case in which the system is fully aware of the possible answers that a user might give in response to a question. Information theory (Shannon and Weaver, 1949) provides a framework that deals very well with this type of scenario. The general idea is that, the less a system initially knows about the actual value of a particular variable, the more benefit there is to inquiring about that variable.

If a question has possible answers v_i, and if the probability of each of these answers is represented by $P(v_i)$, then (according to Russell and Norvig (1995)) the information-theoretic value of obtaining the answer is

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

The information content is measured in bits. For example, suppose we are considering asking a question with six possible answers. If we have no information about the likelihood of each of these answers, we would assign a probability of 1/6 to each. By actually asking the question, the information value is

$$I\left(\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}\right) = 6\left(-\frac{1}{6}\log_2\frac{1}{6}\right) = 2.58 \text{ bits}$$

.

However, if we believed ahead of time that one of the answers had a probability of 2/3, while each of the other five answers had probability 1/15, the information value of the question would be only

$$I\left(\frac{2}{3},\frac{1}{15},\frac{1}{15},\frac{1}{15},\frac{1}{15},\frac{1}{15}\right) = \left(-\frac{2}{3}\log_2\frac{2}{3}\right) + 5\left(-\frac{1}{15}\log_2\frac{1}{15}\right) = 1.69 \text{ bits}$$

.

There is less of an information gain than in the previous example because we already had a good idea of what the response would be, before we even asked the question. As a more extreme example, if we believed answer A to have a probability of 0.99, and each of the other five answers 0.002, the information value would be only 0.104 bits.

If probabilities are not available, but if the system still has a finite set of possible answers, then it can simply assume equal probability for all answers until further evidence is obtained.

Information theory is used quite often in decision tree learning for deciding which attribute to choose at each step when trying to classify a set of examples. For our purposes, the idea behind using information theory is to give us a measure of the system's uncertainty about a particular variable, by measuring how much

information would be gained by obtaining the definite value of that variable.

In applications where systems do not have information about the possible answers, or where the set of possible answers is infinite, an alternative to this information-theoretic measure of uncertainty will be needed.

In determining what they would do next *without* further user assistance, some systems (e.g., (Kozierok, 1993)) might return a *confidence level* in the success of the proposed action. If available, this confidence value can be used to measure the system's uncertainty directly. If $CONF$ is the system's confidence in its next action, then the uncertainty level can be set to $1 - CONF$.

In the absence of such a confidence value, the system will have to resort to using a heuristic capturing its general uncertainty about its domain knowledge. Such a heuristic should take into account the system's experience with working in the domain (the number of problems it has solved in this domain, the number of "similar" instances it has seen), the system's knowledge of this user (how many times it has interacted with this user, along with a sense of how many times it should have interacted, in order to really be sure of its user model), as well as any information the system has about its performance on similar tasks in the past (based on user feedback or the ability to observe directly the success of a task). We discuss this idea further in Section 6.1.6, but for now, we will focus on the information-theoretic case, where the possible answers to a question are known in advance.

Importance

Even if a system has a great deal of uncertainty about a particular topic in its domain of interest, however, it is quite possible that this information could turn out to be irrelevant in the system's decision-making process. For example, a system might have no idea about the user's preference between the movies being shown on two different flights, but if it is believed that acquiring this knowledge would have little or no impact on the user's ultimate choice of flight, then there is no need to ask for the information.

To deal with this issue, we introduce the idea of the *importance* or *criticality* of a particular variable or question in the domain. It would be up to the system designer to assign criticality values to each of the possible topics and subtopics.

In many domains, the value of a problem solution might be measured by means of a formula that includes weights on a number of criteria. For example, in a travel domain, the value of a final travel plan might be measured by a weighted combination of measures of how well the plan satisfies the user's preferences on issues such as the preferred airline, preferred time of day, etc.

In such a scenario, these weights could be used as guidance in representing the importance of each component part. The importance of asking the user about his airline preference, for example, is directly related to the weight associated with that topic in the evaluation function.

When such an evaluation function is not available, system designers will have to assign importance values to each topic by hand. One argument against this approach is that these criticality values are very subjective. However, we would

argue that these values are no more subjective than utilities that would be elicited from users or system designers in a full-fledged utility-based system. In fact, it should be easier for people to assign single criticality values to specific classes of information than to assign abstract utility values to complex outcomes whose worth depends on a large number of attributes. Using the travel domain as an example again, a system designer would have a fairly good sense that, for most users, certain aspects of a flight description (cost, time of day, airline, etc.) are going to be more important than others. The importance values could be specified without knowing the exact preferences of any given user, but knowing only that certain attributes are more likely to influence these preferences.

We advocate that importance values be elicited on a qualitative scale (from very high to very low) and then converted to an appropriate numerical scale.

The level of granularity at which these criticality values are assigned is again up to the system designer, as part of the process of deciding on a task representation. We would recommend that *classes* of topics be assigned an overall criticality level, but that it should be possible to override this value by specifying importance values for particular questions that might arise within those classes.

In most domains, although the initial importance values would be set by the system designer, users should have the opportunity to adjust these values as they deem appropriate.

User knowledge

The preceding discussion leads to the following preliminary formula for a heuristic for the value of interaction:

$$\text{Value of question} = \text{Importance} \times \text{Uncertainty}$$

However, as in our earlier models, this value must be tempered by the fact that the user might be unable to provide an answer to the question. This is where P_{UK}, the probability of the user having the knowledge to respond to a question, comes into play. A particular fact could be important and a system could be highly uncertain about it, but there is little point in asking the user about this fact if we believe that he is almost certain not to know the answer.

Finally, we must introduce a constant term κ to the formula, to ensure that benefits and costs are on the same scale. The value of κ must be determined experimentally. The formula then becomes

$$\text{Value of question} = \kappa \times P_{UK} \times \text{Importance} \times \text{Uncertainty}$$

We will talk in more detail about P_{UK} and how its value should be extracted from a user model in Section 3.8.

To illustrate this alternative model, consider again the simple path-choosing example from Section 3.3.2. The question that was being considered in that example was a simple question with only two possible answers and with the system

believing that those two answers were equally likely. Therefore, we can use the information-theoretic approach to measuring uncertainty. The information content of the question is:

$$I\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1 \text{ bit}$$

.

As specified in the original description of the example, the probability of the user actually knowing the correct answer to the question is 0.6.

The final components of the formula are the importance of the question and the constant value κ. In this case, the answer to the question is fairly important, but it is definitely possible for the system to proceed without the user's help. Suppose that the importance of this question has been set at 40 out of 100. Suppose also that the system has learned that an appropriate value of κ if 0.4. Then,

$$\begin{aligned}\text{Value of question} &= \kappa \times P_{UK} \times \text{Importance} \times \text{Uncertainty} \\ &= 0.4 \times 0.6 \times 40 \times 1 \\ &= 9.6\end{aligned}$$

Table 3.7 shows the same variations on the path-choosing example that were included in Table 3.4. Costs are computed in exactly the same way as in Section 3.3.2, and so the cost values are simply copied from the earlier table. However, the benefits are now computed by using the above formula for the value of asking the question, rather than by considering the expected utilities of the course of action that would be taken if the system did or did not interact. If the two tables are

compared, it can be seen that the decisions made by the two agents are identical. However, the agent described in this section accomplishes this without expensive reasoning about future sequences of events.

P(U knows)	**P(Path 1 clear)**	t	b	**Inf. content**	**Benefits**	**Costs**	**Ask?**
0.6	0.5	10	10	1	9.6	5	Yes
0.2	0.5	10	10	1	3.2	5	No
0.6	**0.9**	10	10	0.47	4.5	5	No
0.6	**0.1**	10	10	0.47	4.5	5	No
1.0	**0.9**	10	10	0.47	7.5	5	Yes
0.6	0.5	**50**	10	1	12	13	No
0.6	0.5	10	**40**	1	12	14	No

Table 3.7: Variations on path-choosing example revisited

3.6.2 Examples

We will now consider two examples from the travel domain to demonstrate how the heuristic model works.

Example 1: Suppose we are trying to decide whether or not to ask the user for the preferred time of day for a flight (assuming it was not specified initially).

This is an example of the case where we know the possible answers (and their probabilities), so we can use the information-theoretic approach to measuring uncertainty.

Suppose that flight times are divided into four time segments (morning, afternoon, evening, overnight), and that we expect the following probability distribution for the user's preferred time segment: $P(Morning) = 0.15, P(Afternoon) =$

$0.50, P(Evening) = 0.30, P(Overnight) = 0.05$. In this case, uncertainty is computed using the formula for information gain as $I(0.15, 0.50, 0.30, 0.05) = 1.65$.

The importance of this question is quite high (75). Since almost all questions in the travel domain will be about the user's own preferences, we will assume that P_{UK} will be 1.0.

Benefits = κ x P_{UK} x Importance x Uncertainty

$= 0.4 \times 1.0 \times 75 \times 1.65$

$= 49.5$

This is very high, but this is intuitive: we *should* be very likely to ask this question.

For costs, we will assume low cost values for time (20) and for bother (5), since the question is quite straightforward and should be understood by the user to be highly relevant.[22] Assume $w_t = 0.4$, $w_b = 0.3$. Then Costs = 0.4(20) + 0.3(5) = 9.5. Benefits outweigh costs, so we will ask the question.

Example 2: Instead of asking about preferred time of day, suppose we are considering asking about the selection of magazines available on the flight. The importance for this specific topic should be very low. Suppose it is set at 5 (out of 100).

For the possible answers, let us assume that we are asking users to choose their

[22]Note that we are assuming that the question will actually be reasonable – for example, that the system would not present the user with a certain time range as an option if there are not actually flights available at that time. If this type of question were possible, then the "bother" experienced by the user upon discovering that an irrelevant question was asked might be significantly higher.

two favourite "types" of magazines from a list of ten. Only some subset of these types are available on each airline. There are 45 possible combinations of two magazine types; suppose we have a prior probability distribution such that the information value for the question is 2.65.[23] P_{UK} is still 1.0.

$$\text{Benefits} = 0.4 \times 1.0 \times 5 \times 2.65$$
$$= 5.30$$

Costs: Suppose the time and bother costs are estimated as being 30 and 5, respectively, with the same weights as above.

Costs = $0.4(30) + 0.3(5) = 13.5$.

Costs are higher than benefits, and we would not bother asking this question.

This section has provided an alternate approach to computing the benefits of interacting with a user, without projecting ahead to consider all possible future sequences of actions, their consequences and utilities. Instead, it uses information about the system's current uncertainty about its knowledge, the perceived importance of the type of question being considered and the probability of the user being able to provide assistance.

Once these benefits are computed, the decision procedure is the same as it was in earlier models. The costs are computed in exactly the same way as shown in Section 3.3.2, and the system will decide to interact if the benefits exceed the costs.

[23] This comes from an example distribution where most of the combinations of magazine types have probability 0, but 8 of the 45 combinations have some positive probability.

3.7 A design procedure

In this section, we will provide a systematic approach for the design of a mixed-initiative system that makes use of the models presented in this thesis for reasoning about interacting with users. We will first discuss the conditions under which each of the different models is appropriate. We will then provide a procedure for designing such a mixed-initiative system. This procedure will include a list of all the factors that must be modeled and when each should be provided by the system designer or obtained from the user.

3.7.1 Choosing the appropriate model

Figure 3.8 illustrates a decision process for choosing the appropriate model for a given application area. The details of this process will be discussed in the following paragraphs.

The single-decision model presented in Section 3.3.2 is applicable only when the domain is one in which all decisions are one-shot decisions: the system has to choose an action to perform, will complete the entire task after making that single choice, and must decide whether to gather more information from the user before making the decision. This single-decision model can be extended to deal with simple sequential decision situations using the dynamic programming approach discussed briefly in Section 3.3.3. However, this applies only when all possible state sequences are of finite length.

For the case of infinite possible sequences of events, a well-known extension to dynamic programming uses the Markov decision process. This approach, discussed

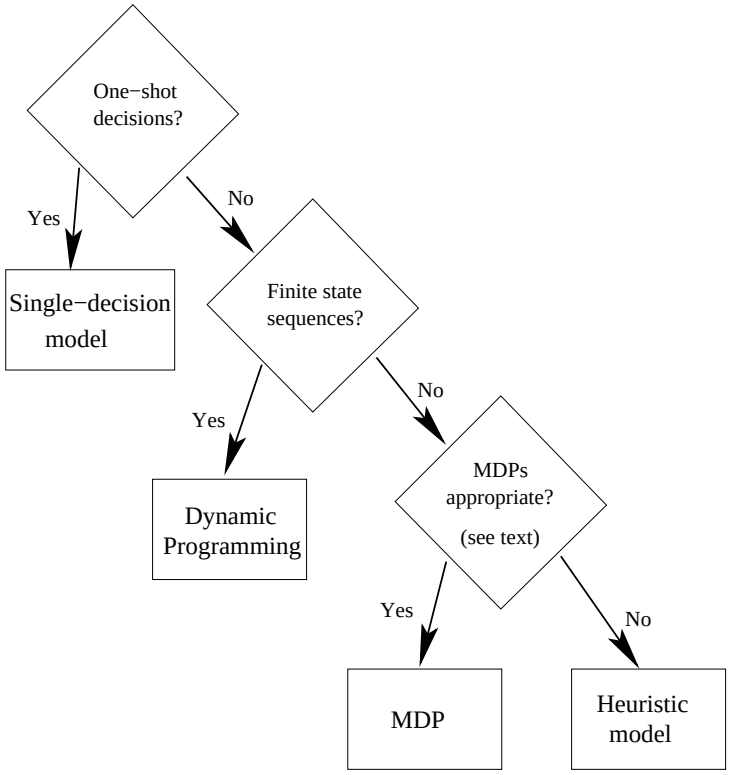


Figure 3.8: A decision procedure for choosing the appropriate model

in Section 3.5, provides an exact and systematic technique that has been shown to be effective for rational decision-making in many different domains (Feinberg and Shwartz, 2001).

When it comes to applying the MDP approach to *dialogue* with users, however, there are a few obstacles. We suggest the following checklist for system designers to use in determining whether an MDP approach is appropriate for their problem of interest. If all of the statements below are true, then such an approach should be successful. Otherwise, an alternative model, such as the one presented in Section 3.6, will be required.

1. It is possible to construct a reasonably-sized state space, possibly by representing the current state with a small set of finite variables.

2. A finite set of actions can be specified, including actions involving interaction with the user.

3. It is possible to determine, for each state/action pair, the probabilities of transitions to each possible next state.

4. It is possible to identify all final outcomes to problem-solving sessions and to specify rewards for all of them, as well as rewards (possibly negative) associated with every state or state/action pair. (This might come in the form of a utility function that assigns utility values to states by evaluating the states on the basis of a number of subutility functions.)

5. The state is accessible. The system will always know exactly what state it is in at any time.

6. It is reasonable to recompute the optimal policy frequently, OR the models of the user and domain are not expected to change significantly or frequently as the system gains experience.

If all of these points are satisfied, then the designer should refer to a standard MDP reference (e.g., (Puterman, 1994)) in order to build the appropriate system: establishing the state space, the set of actions, the transition model and the reward model. The main contribution of our work in this case is the identification of dialogue-specific factors such as the user's willingness to interact and the user's

perceived knowledge of the domain to be used in building the transition model and reward (cost) model.

If item #5 on the list above (the accessibility of the state) is the only obstacle, then it will be possible to represent the problem as a partially-observable MDP, which can then be approximated by a dynamic decision network (DDN). If the DDN approach is chosen, then an explicit state space will not be required; however, a set of state variables (as described in Section 3.5.4) will have to be established. These state variables will allow the system to use whatever evidence it has available to determine a probability distribution on the current state. The dynamic decision network method is appropriate as long as the designer is comfortable assigning conditional probabilities to the nodes representing the state variables and assigning utilities to the different situations that might arise.

If neither the MDP approach nor the dynamic decision network approach is deemed appropriate, then the heuristic model discussed in Section 3.6 will be useful. In this model, information about the possible consequences of every action and the utilities of possible outcomes many steps in the future are not required. This analysis is instead approximated by considering how uncertain the system is about the question it is considering asking the user, how important that question is considered to be, and how likely it is that the user will be able to provide a helpful response.

Table 3.8 shows a list of all of the numerical factors discussed in this chapter and whether each one is necessary for each of the possible models that have been considered.

Factor	Single-decision	MDP	Heuristic
Utilities $U(s)$ for all s	x	x	
Importance of questions Imp			x
Overall Task criticality $Taskcrit$	x	x	x
P_{UK}	x	x	x
Initial beliefs about the world	x	x	x
Time required for each type of interaction	x	x	x
Time cost function t	x	x	x
Bother cost function b	x	x	x
Other cost functions C_i	x	x	x
Weights on cost factors w_i	x	x	x
Function for measuring uncertainty Unc			x
User willingness (w)	x	x	x
User preference information	x	x	x

Table 3.8: Numerical factors included in models of Chapter 3

3.7.2 Steps in System Design

Below is a list of important steps that must be followed by a system designer wanting to use the models presented in this thesis. The steps are divided into three categories: (1) issues that must be addressed when the designer is first implementing the system, (2) steps that must be taken when each user first uses the system, (3) updates that must occur as the system interacts with users.

For several of the steps mentioned below, we have made suggestions throughout the thesis as to how exactly the issues might be modeled. However, the overall model is flexible enough to allow alternative methods or algorithms to be substituted for our own recommendations.

Unless otherwise indicated, all steps are relevant for all models discussed in this

chapter. We first present a summary of the steps involved in constructing the system, to be followed as a checklist for the design procedure. This is followed by a detailed discussion of each point in the checklist.

Summary

Initial system design:

- Establish the problem representation.
- Design the user interface.
- Establish utility/reward functions. (For the information-theoretic heuristic model, establish importance values for different questions in the domain instead.)
- Specify the task criticality value for the domain.
- Determine the exact role of specific user knowledge, stereotypical user knowledge and knowledge expected of all users.
- Incorporate understandability.
- Provide the system with an initial set of beliefs about the application domain. Establish how those beliefs will change according to new information that the system gathers (from the user or otherwise).

- Determine the role of time and time criticality. Provide initial estimates for the amount of time that each action will take, and specify how those estimates will change as the system gains experience.
- Address the issue of bother cost, specifying how the bother will be measured as a function of the user's willingness. and establishing how the bother function depends on the number of interactions so far in the dialogue and on the recency and estimated cognitive load associated with each.
- Determine if any other cost factors are relevant for the domain of interest. Establish initial weights for the cost factors.
- If desired, establish a set of user stereotypes.

When user first employs system:

- Assign the user to a stereotype.
- Establish the user's willingness to interact.
- Elicit any needed preference information from the user.

Updates during/after user sessions:

- Update the system's beliefs about the world, as new information is observed.
- Update the representations of various types of user knowledge information.
- Adjust the user's willingness to interact, via learning or explicit user feedback.
- Maintain the dialogue history.

- Adjust the weights on cost factors, via learning or explicit user feedback.

Detailed discussion

We now discuss each of the points in the above list in more detail.

Initial system design

- Establish the representation that will be used for the problem(s) involved in the application domain. How is the task represented? How is a state of the world represented? What actions can the system take? What is the transition model between states? (If a particular action is taken in a particular state, what new state(s) will result and with what probability?) What constitutes a solution to the problem? How are tasks broken down into subtasks?

- Design the user interface for the system. This is not our focus in this thesis, but it is still an important concern. Particularly relevant to our work is the issue of how the user is able to take control of the application. What mechanisms will exist to allow the user to take an action? Are there restrictions on which user actions are allowed in different situations?

- Establish how the utilities of states (or the rewards associated with solutions to the task) will be computed. This involves specifying values $U(s)$ for all possible states s, or a function that allows these utilities to be computed by evaluating each state according to a set of criteria. This will be a domain-dependent exercise: to decide what factors will be combined to determine the value of a solution. For example, in a scheduling domain, a schedule might

be evaluated according to the constraints that are violated by the schedule and the penalties associated with violating each of those constraints.

In many cases, utility functions will be user-dependent: the exact utilities of each outcome will depend on the preferences of an individual user. This issue will be addressed in a later step; however, at this step, the system designer should specify *how* the utility function will depend on any user-specific factors.

For the information-theoretic heuristic model of Section 3.6, information should be specified not about the utilities of states, but instead about the importance Imp of different questions or classes of questions that might come up in the problem solving. This importance value is intended to reflect how crucial a role this particular aspect of the task is expected to play in evaluating the overall quality of a problem solution.

- Establish the task criticality value $Taskcrit$ for the domain as a whole. For example, a domain such as military planning would have a far higher task criticality value than a problem in which a system is recommending menu items at a restaurant.

- Determine the role of modeling user knowledge information. There are several options available to the system designer, as shown in Figure 3.2. The designer must decide, for example, whether she wishes to model information about how strong each specific user's knowledge is perceived to be in each specific area of expertise within the domain, or more general knowledge about what all users (or all users of a particular "type") might be expected to know about

the domain in general.

Most likely, a system designer will want to use some combination of the various types of user knowledge modeling mentioned in Figure 3.2. In this case, there must be some mechanism in place for managing the different types of information and for updating values as new information is obtained. In particular, how is the value of P_{UK} generated at any given time, based on the system's user modeling information? For example, how is stereotypical information about the expectation of the *general* user's knowledge merged with specific information about this user's knowledge? Similarly, what is the role of the system's model of users' knowledge about the domain in general and about specific topics? We provide some suggestions for how to manage these different models in Section 3.8, but the model allows for alternate approaches to be substituted.

- Provide a mechanism for the system to use to keep a record of the dialogue so far – either a dialogue history (as seen in Collagen (Rich and Sidner, 1998)) or, at the very least, an itemized list of interactions with the user, their perceived "cost" and *when* they happened.

- Provide the system with an initial set of beliefs about the world. If the task representation consists of a set of random variables that can take on any of a set of possible values, then the system's beliefs are represented as probability distributions over the possible values for each random variable. For example, in the path-choosing example, the status of path 1 would be a variable (with possible values 'clear' and 'busy'). If the system initially has no knowledge of

the status of path 1, then it would assign a probability of 0.5 to each of those values.

Determine how these values should be updated when the system observes new information or when the user provides information about a variable. For example, if the user says that $x = 5$, then what should the system do if it previously believed that $x = 4$ with probability 0.8 and $x = 5$ with a probability of only 0.2? Does it take the user's word and make $x = 5$ a "known" fact? Does it revise its beliefs according to some standard formula?

- Establish the role of time. What is the penalty function associated with time elapsing? In other words, how does the *value* of finding a solution decrease – if at all – as time elapses? This involves specifying a function $t(s, a)$ indicating how the time penalty depends on the current state and on the particular action (e.g., question for the user) being considered.

  As mentioned in Section 3.4.3, this time cost function will likely require values for two variables: c (or β), indicating how time-critical the problem is, and T, an optional specification of a deadline after which no solution will have any value. Defaults should be provided for these values by the system designer, with the option available for subsequent adjustment by users.

  Also related to time, the designer must provide initial estimates for the amount of time that each type of action (including interaction) will take. This information does not necessarily have to be provided as an exact number of seconds or minutes, but can be specified on a qualitative scale. For example, in the travel domain, the time requirement for asking the user to

rank all available airlines might be listed as very high, whereas the time requirement would be low for a simple question about the user's preference between aisle and window seats. These qualitative descriptions can then be converted to a normalized 0-100 scale, in order to obtain a time cost penalty to be used in the formulas.

The system designer might also want to specify how these estimates could be revised as new observations are made during problem-solving sessions with a user. If the designer proves to be wrong with her initial guess at the length of time required to interact with a user, it is important for the system to be able to adjust this estimate through learning techniques.

- Specify the bother cost function b. Determine how the bother will be measured as a function of the user's willingness as suggested in Section 3.4.3. Establish also how the bother function depends on the number of interactions so far in the dialogue and on the recency and estimated cognitive load associated with each. Again, we provide recommendations in Section 3.4.3, but system designers are free to use any function for estimating the cost of bothering the user.

  Provide a mechanism that allows for the user to adjust the willingness level (which will be set when the user first employs the system, as discussed below).

- Determine if there are any other cost factors C_i involved in the domain of interest: resource costs, etc.

  Assign initial weights w_i to the various cost factors, based on the system

designer's intuition or on empirical results about the role of each relevant cost factor. Provide a mechanism by which the user – and possibly the system (through learning and user feedback) – can adjust the weights. Ensure that weights are chosen so that Benefits and Costs end up on the same scale.

- If using the heuristic model of Section 3.6, determine how uncertainty will be measured from among the alternatives discussed.

- If desired, develop a set of user stereotypes and a set of criteria to be used in determining to which stereotype each user belongs, based on an initial interview.

When user first employs system

- Assign the user to a stereotype, based on an initial interview. For instance, a user's general expertise might be expert, intermediate or novice, according to his own self-evaluation or according to his responses to a questionnaire.

- Establish the user's willingness to interact with the system. Have the user estimate this value on a quantitative or qualitative scale. This process could be assisted by asking the user a standard set of questions to establish how active he wishes his role to be in the problem solving.

- Elicit any preference information from the user that is needed to solidify the system's model of the utilities of different possible outcomes in the domain.

Updates during/after user sessions

As the system interacts with users, many components of the system's user model and task model will have to be adjusted.

- The system's beliefs about the world might change, based on new observations. This is not our focus in this work, but there is an entire body of research on belief revision and nonmonotonic reasoning (e.g. (Bacchus et al., 1994)).
- As the system observes whether or not the user was able to answer a particular question, it will have to adjust its representation of the user's knowledge.
- The user's willingness to interact might be adjusted explicitly by the user, or automatically by the system as it receives feedback on the actual bother experienced by the user.
- The dialogue history will have to be maintained. This includes storing information about each interaction with the user, the time at which it occurred, and the perceived cognitive cost of the question involved.
- The weights on cost factors might be adjusted. Again, this could be done explicitly by the user or automatically by the system, as it learns from its performance.

This section has provided a systematic procedure for designers to use in developing mixed-initiative systems that make rational decisions about interacting

with users. The process has been broken down into steps to follow during the initial design stage, during the first interactions with each individual user and during subsequent typical user sessions.

3.8 User modeling

As mentioned in Section 3.2, a user model is a system's internal representation of a user's knowledge, abilities, preferences, goals and any other user-specific information that helps a system to adapt its behaviour in a way that is appropriate for each user. The user model is a crucial component of our model for reasoning about interaction with users. In particular, we have emphasized the importance of reasoning about the user's perceived knowledge and about the user's willingness to interact with the system over the course of a session.

As a system interacts with different users, it will often have to update its user models to reflect the new evidence that it has acquired. For example, a system that initially believed a user to be knowledgeable about a particular topic, but discovered that the user was twice unable to answer questions about that topic, should update its beliefs about that user to reflect this new information.

3.8.1 User Willingness

For the most part, the user should be primarily responsible for updating the variable representing his willingness to be an active participant in dialogues with the system. The user will be able to rely on his intuitive sense of how satisfied he has been with dialogues with the system in order to gauge whether or not the system is too

bothersome. For example, a user who believes that the system is interrupting him far too often to ask mundane questions can access his profile and adjust the bother parameter to indicate that he wishes to be consulted less frequently.

This introduces an issue that is an important ethical one in all user modeling research: should users always have the ability to view and/or edit the information that the system stores about them (Cook and Kay, 1994)? With the possible exception of highly critical applications where user modeling data is extremely reliable and has been carefully collected, we believe that users should always have this ability. However, users should be made aware of how the changes they are making will affect the system's behaviour.

Depending on the sophistication of the system, it might be possible for it to use certain cues to initiate a change to the bother factor. For example, in the unlikely case that the system is equipped with vision capabilities and enough intelligence to detect frustration on a user's face, this might be used to recognize that the user's bother level should be adjusted. Similarly, a system that is able to detect that the user is particularly busy at a given time might take the initiative to adjust the user's bother level temporarily.

3.8.2 User Knowledge

The aspect of the user model that is more important for the system to maintain is that of user knowledge. As mentioned in Section 3.2.2, in determining the likelihood of a user knowing the answer to a particular question, a system might want to incorporate different types of information about user knowledge. This includes

information about the *specific* user's knowledge, about the knowledge expected of users in the same *stereotype*, and about the general knowledge expected of *all* users. For each of those classes of information, a system might have expectations about the probability of getting an answer to the *specific* question being considered, about a particular *class* of questions, or about *any* question at all in the domain.

In this section, we provide a matrix-based framework that can be used to determine, at any moment in time, a single P_{UK} value representing the probability of the user being able to answer a particular question. This is done by considering all of the types of information described in the previous paragraph, weighted according to the amount of experience the system has with particular users and topics and according to initial weights specified by the system designer.[24] We will describe how information about user knowledge is stored, how it is used to compute P_{UK}, and how it is updated as the system makes new observations.

Storing user knowledge information

Suppose that a system is aware of m distinct users and n stereotypes to which users might belong. Suppose also that we are able to enumerate i possible questions or topics that a system might want to ask of a user, and j classes into which these questions can be categorized.

Our framework then requires the following data structures:

[24] The idea of considering a single user, user stereotypes and all users is in the same vein as the concept of weighted combinations, used in statistical natural language processing (Chi et al., 2001).

- For each user, construct a $1 \times (m + n + 1)$ vector **u**. The first m entries are used to identify the user as one of the m users about which the system has a profile. These m entries include a single 1 and $m - 1$ zeroes. In the example vector below, the user is the second of four known users. The next n entries classify the user into one or more stereotypes. If the system designer wishes for each user to belong to only one stereotype, then these n entries will include a single 1 as the only non-zero number. However, it is also possible to indicate that a user possesses some characteristics of more than one stereotype by including multiple non-zero entries that sum to 1. In the example below, there are three stereotypes; the user is primarily identified as belonging to the first stereotype, but has some properties of the third as well. The final entry of this vector is always 1 and indicates that the user is a member of the class of "all users" of the system.

$$\left(\begin{array}{cccc|ccc|c} 0 & 1 & 0 & 0 & 0.9 & 0 & 0.1 & 1 \end{array} \right)$$

  Also for each user, construct a second $1 \times (m + n + 1)$ vector **uw**. This vector stores weights to indicate how much impact each *type* of user-knowledge information should have in computing P_{UK}. Initially, the system might have no information at all about individual users, in which case the weight on the user-specific information should be zero.[25] In these early situations, predictions about the knowledge of a user will be based entirely (or mostly)

[25] In some systems, initial interviews might be used to gather some concrete information about individual users, in which case some weight would be placed on the user-specific information that is gathered.

on stereotypical information and estimates of the knowledge expected of *any* user. The actual weights for stereotypes and for the "all-users" entry should be set up according to the desired contribution of each. For instance, if the system designer is confident that a user's stereotype will be a very accurate predictor of the user's behaviour, then she might set the initial weight on the stereotypical information to be quite high relative to the weight of the "all-users" entry (say, 0.9 vs. 0.1), as shown in the example vector below. If, on the other hand, there are no stereotypes defined or there is low confidence in their relevance to any given user, the information about all users should carry most of the weight.

$$\left(\begin{array}{cccc|ccc|c} 0 & 0 & 0 & 0 & 0.9 & 0.9 & 0.9 & 0.1 \end{array} \right)$$

The weights in this vector will be updated as the system makes its own observations about the knowledge of users, again according to the specifications of the system designer. For example, once the system has observed the user's expertise on a particular topic on a few occasions, this user-specific information should be the primary predictor of whether the user will be helpful on this topic in the future. In other words, the weight of the contribution from the user-specific information should increase gradually until it eventually overwhelms the weight on the stereotype and "all-users" contributions. An example of this, and further discussion on how to update the weights, will appear later in this section.

- For each possible topic or question, construct two $(i + j + 1) \times 1$ vectors $\mathbf{t}$

and **tw**. These are analogous to the user vectors described above. In the first vector, the first i entries will uniquely identify the question, the next j entries will classify it into one or more classes of questions, and the final entry will always be one. The example below shows the topic vector for the third of five questions, classified as belonging to the second of two question types.

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ -- \\ 0 \\ 1 \\ -- \\ 1 \end{pmatrix}$$

The second vector **tw** will indicate the desired weights on specific questions, general topic areas, and the domain as a whole.

- Construct a large $(m + n + 1) \times (i + j + 1)$ matrix **PK**. This matrix stores probability values: each entry represents the likelihood that a specific user, a class of users, or any user at all would know the answer to a specific question, a question of a certain type, or any question in the domain. The rows identify the user or user stereotype, while the columns represent questions or topic areas. In the matrix below, for example, the bold entries indicate that the

first user has a probability of 0.8 of being able to answer the first question, and that users belonging to the third stereotype have a probability of 0.7 of answering questions of the second type.

$$\left(\begin{array}{ccccc|cc|c}
\mathbf{0.8} & 0.6 & 0.5 & 0.2 & 0.0 & 0.7 & 0.3 & 0.7 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.7 & 0.3 & 0.6 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.6 & 0.3 & 0.5 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.6 & 0.3 & 0.6 \\
\hline
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.8 & 0.5 & 0.6 \\
0.8 & 0.7 & 0.8 & 0.2 & 0.0 & 0.6 & 0.6 & 0.7 \\
0.9 & 0.8 & 0.9 & 0.2 & 0.0 & 0.4 & \mathbf{0.7} & 0.6 \\
\hline
0.8 & 0.7 & 0.8 & 0.2 & 0.0 & 0.6 & 0.6 & 0.6
\end{array}\right)$$

Now, if the system is considering whether or not to ask a user a question, it can use matrix multiplication to determine a single P_{UK} value, based on a weighted combination of the above information.

This is done by multiplying **u** by **uw** element-wise, to yield a new vector $\mathbf{u_{wtd}}$, multiplying **t** by **tw** element-wise, to yield a new vector $\mathbf{t_{wtd}}$, and then using matrix multiplication to obtain $P_{UK} = \mathbf{u_{wtd}}\ \mathbf{PK}\ \mathbf{t_{wtd}}$.

In practice, when the vectors and matrices are very large, it will make sense to perform these multiplications not with the entire matrices, but by selecting only the rows and columns that are relevant to the current user and the current question.

This is demonstrated in the following example.

Example 1. Suppose we have three different users, two stereotypes, and four different possible questions that are classified into two topic areas. For this example, we are considering asking user 1 (who belongs to stereotype 2) the third question in the list (belonging to topic area 2). The vectors **u** and **t** are therefore set as follows:

$$\mathbf{u} = \begin{pmatrix} 1 & 0 & 0 & | & 0 & 1 & | & 1 \end{pmatrix}$$
$$\mathbf{t} = \begin{pmatrix} 0 & 0 & 1 & 0 & | & 0 & 1 & | & 1 \end{pmatrix}^T$$

We have no evidence about the specific user's knowledge, so the **uw** vector below shows that we are relying primarily on estimates of all users' knowledge, to a lesser extent on stereotype information, and not at all on specific user knowledge. Note that the probabilities within each segment of the vector are replicated, and the representative probabilities (0, 0.2 and 0.8, in this case) always sum to 1. Similarly, the **tw** vector below shows that our estimates are based heavily on what the system knows about the general knowledge domain of users, as opposed to their abilities to deal with specific questions.

$$\mathbf{uw} = \begin{pmatrix} 0 & 0 & 0 & | & 0.2 & 0.2 & | & 0.8 \end{pmatrix}$$
$$\mathbf{tw} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & | & 0.2 & 0.2 & | & 0.7 \end{pmatrix}^T$$

Finally, the probability matrix **PK** is as follows. The entries with question marks in the first row reflect the fact that we have made no observations about this

particular user, and therefore we have no corresponding probabilities.

$$\left(\begin{array}{cccc|cc|c} ? & ? & ? & ? & ? & ? & ? \\ 0.7 & 0.6 & 0.7 & 0.2 & 0.7 & 0.3 & 0.6 \\ 0.7 & 0.6 & 0.7 & 0.2 & 0.6 & 0.3 & 0.5 \\ \hline 0.7 & 0.6 & 0.7 & 0.2 & 0.8 & 0.5 & 0.6 \\ 0.8 & 0.7 & 0.8 & 0.2 & 0.6 & 0.6 & 0.7 \\ \hline 0.8 & 0.7 & 0.8 & 0.2 & 0.6 & 0.6 & 0.6 \end{array}\right)$$

To perform the matrix multiplication, in order to obtain an overall P_{UK} value, we would select only the nonzero entries from the vector **u** and the corresponding weight vector **uw**. This yields new vectors **u** = (1 1 1) and **uw** = (0 0.2 0.8). Performing the element-wise multiplication of these two vectors gives the result $\mathbf{u_{wtd}}$ = (0 0.2 0.8). Similarly, we would shrink the vectors **t** and **tw** by selecting only the nonzero entries in **t** and the corresponding entries in **tw**: $\mathbf{t} = (1\ 1\ 1)^T$, $\mathbf{tw} = (0.1\ 0.2\ 0.7)^T$. Performing element-wise multiplication yields $\mathbf{t_{wtd}} = (0.1\ 0.2\ 0.7)^T$.

Finally, from the large probability matrix **PK**, we will use only rows 1, 5 and 6 (corresponding to the nonzero entries in **u**) and only columns 3, 6 and 7 (corresponding to the nonzero entries in **t**).

Performing the matrix multiplication yields an overall P_{UK} value of

$$\begin{pmatrix} 0 & 0.2 & 0.8 \end{pmatrix} \begin{pmatrix} ? & ? & ? \\ 0.8 & 0.6 & 0.7 \\ 0.8 & 0.6 & 0.6 \end{pmatrix} \begin{pmatrix} 0.1 \\ 0.2 \\ 0.7 \end{pmatrix} = 0.634$$

Example 2. As a second example, consider the situation where the system is considering asking the same question of the same user at a later time. Because the system has now had more experience in dealing with this user and in asking this question of different users, the weight vectors have changed so that more weight is placed on specific knowledge information. In particular, suppose that **uw** has changed from (0 0 0 | 0.2 0.2 | 0.8) to (0.75 0.75 0.75 | 0.05 0.05 | 0.2), and that **tw** has changed from $(0.1\ 0.1\ 0.1\ 0.1 \mid 0.2\ 0.2 \mid 0.7)^T$ to $(0.6\ 0.6\ 0.6\ 0.6 \mid 0.1\ 0.1 \mid 0.3)^T$.

Also, since the system has now observed the user's ability to answer certain questions, the first row of the **PK** matrix (which contained only question marks before) will now contain actual values: say, (0.8 0.8 0.9 0.9 | 0.8 0.9 | 0.85).

The matrix multiplication will now yield a P_{UK} value of 0.797. This is a significant change from the value 0.634 that was found before. The increase is due to the fact that the system has now observed specific information about this user and has found him to be quite knowledgeable.

Updating probability values

The actual probabilities of different users being able to answer different questions are also stored in a set of matrices. One matrix $\mathbf{N_{ask}}$ maintains a record of the number of times each question or type of question has been asked of each user or type of user. An example matrix is shown below, where there are five different

users, three different stereotypes, six different questions and two different "types" of questions. For instance, row four shows that User #4 has been asked the first question 3 times and the fifth and six questions once each. All of these questions fall under the second question type. The final entry in the row shows the total number of questions that have been asked of the user.

Similarly, column 4 shows that question #4 has been asked once of user #1 and twice of user #3, once of users in the first stereotype and twice of users in the second stereotype, and 3 times in total.

$$\mathbf{N_{ask}} = \left(\begin{array}{cccccc|cc|c} 2 & 2 & 4 & 1 & 0 & 0 & 7 & 2 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 0 & 6 & 4 & 7 & 11 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 & 5 & 5 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 5 & 2 & 4 & 1 & 1 & 1 & 7 & 7 & 14 \\ 1 & 1 & 1 & 2 & 0 & 6 & 4 & 7 & 11 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 6 & 3 & 6 & 3 & 1 & 7 & 12 & 14 & 26 \end{array}\right)$$

A second matrix, $\mathbf{N_{ans}}$, contains information about the number of times that each user actually provided an answer to each question.

$$\mathbf{N_{ans}} = \left(\begin{array}{cccccc|cc|c}
1 & 2 & 3 & 1 & 0 & 0 & 6 & 1 & 7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 2 & 0 & 1 & 3 & 2 & 5 \\
3 & 0 & 0 & 0 & 1 & 1 & 0 & 5 & 5 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
\hline
4 & 2 & 3 & 1 & 1 & 1 & 6 & 6 & 12 \\
1 & 0 & 1 & 2 & 0 & 1 & 3 & 2 & 5 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
\hline
5 & 2 & 5 & 3 & 1 & 2 & 10 & 8 & 18
\end{array}\right)$$

The values in the matrices $\mathbf{N_{ask}}$ and $\mathbf{N_{ans}}$ are used to compute the individual values stored in the matrix **PK**. For example, the first user has been asked the first question twice and has answered only once. Therefore, the corresponding entry in the **PK** matrix would be $1/2 = 0.5$.

Updating weights

A final set of vectors is used to maintain the weights on the various components of the model (stored in the vectors **tw** and **uw** discussed earlier). These vectors are constructed as follows.

Initially, assuming that no information is available about specific users, the system designer must make a decision on how much weight should be placed on

information about a user's stereotype and how much weight on information about all users in general. The idea is to decide how many interactions with a single user will put us in a situation where we wish to attribute equal weight to the observed user-specific information and to the prior beliefs about stereotypes or about all users.

For example, we might decide that, once we have interacted with a user 20 times, we have learned enough about that user to put the same weight on user-specific information as we have on information about all users. In this case, the initial weight assigned to information about all users is 20.[26] The same would be done to assign an initial weight to information about the user's stereotype. Let us assume an initial weight of 5.

These decisions are then used to construct a weight vector for each user. Using the example above, this initial vector would be (0 5 20). The 0 in the first entry represents the number of questions that have been asked of this user, and only this entry will change as the vector is updated over time. This vector is used directly in computing the values of the vector **uw** described earlier. First, the vector is normalized so that the values sum to 1; for example, (0 5 20) becomes (0 0.2 0.8). Then, to construct the **uw** vector, the first entry is repeated m times (for the m users known to the system), and the second entry is repeated n times (for the n known stereotypes). Referring back to **Example 1** from earlier in this section, the situation with three known users and two known stereotypes would yield a **uw**

[26] The number of interactions that determines when we have learned enough about a user should really depend on the variance in the user's performance; however, setting a fixed number is a practical solution for the purposes of this thesis.

vector of (0 0 0 | 0.2 0.2 | 0.8).

The idea is for the weights on the more general information to be high initially, but as more information is gathered about the specific user, the value in the first entry will begin to overwhelm the pre-defined weights in the second and third entries. For example, once the system has interacted with the user 25 times, the three-valued vector would become (25 5 20), which would yield a **uw** vector of (0.5 0.1 0.4). After 75 interactions with the user, **uw** would be (0.75 0.05 0.20).

Exactly the same type of process applies to maintaining weight matrices for questions instead of users.

Closing comments on modeling knowledge

We argued earlier in this chapter that it is very important for a mixed-initiative system to maintain a model of the likelihood that users will possess certain types of knowledge. In this section, we have provided a concrete quantitative approach to merging information that the system has available about the knowledge of specific users, of different classes of users and of all users in general, as well as about different classes of questions that might be asked. This type of a hybrid method is a novel approach to modelling knowledge about users and will be of interest to the user modeling community.

Chapter 4

Examples

In this chapter, some examples will be presented to illustrate the use of the models in Chapter 3. In Section 4.1, a simple example application is traced through all of the steps in the design process of Section 3.7 and then through a few sample interactions with users. In Sections 4.2-4.4, we focus on specific parts of the models, and illustrate their applicability to some more realistic, complex problems.

4.1 Example of the Design Procedure

The discussion in this section will illustrate the design process of Section 3.7. Before beginning the example, it should be pointed out that the problem described in this first section is quite simple and straightforward. This choice was made in order to make the illustration of the process clearer and to prevent the reader from being overwhelmed by details. Also, some readers might believe that, in the example application described in this section, it would make sense for the system to

ask for the user's help all the time. The questions that the system might consider asking the user are simple, not time-consuming, and should not cause very much inconvenience to the user at all. Again, the choice of domain was made for the sake of simplicity and readability, and not to illustrate the full power of the models.

This section deals with the following example. An intelligent agent is trying to decide whether to buy one of a set of used books on a user's behalf. In order to make this decision easier, the system is considering the possibility of asking the user one or more of a set of questions. These include such questions as: (A) Do you own book X? (B) How would you rate book Y? (C) Do you prefer mysteries or biographies? (D) What is the most you are willing to let me spend on a book without explicit consultation?

Based on the system's beliefs about the domain, it tries to make a somewhat informed decision about the optimal action to choose for a particular user. However, by asking questions, the system could improve its knowledge and therefore increase its certainty in its decision.

We will first describe the variables in our model that must be set by the system designer before any users are interacting with the system at all, as described in Section 3.7.

4.1.1 Initial system design

Problem representation: Generally speaking, we are talking about the idea of purchasing items that are up for sale. Each available item would be represented

by a series of features. In the case of the books described in our example, these features would include details such as the title, the author, the genre, the price, the condition of the book, the original price of the (new) book, the specific edition and the time at which the sale will be over. Figure 4.1 shows an example of how one particular item would be stored.

Item #	Title	Author	Genre	Price	Cond.	New price	Ed.	Sale ends
2304291	The Game	Ken Dryden	Sports	10.00	Exc.	19.95	3	2003-06-10 17:30

Figure 4.1: Sample entry for used book example

The system would also maintain a database that contains any ratings that have been assigned to books, genres or authors by each user. This database would be consulted by the system to estimate how popular a particular book is with the general user population or with a particular subset (stereotype) of users. All of this would be done to help the system predict how likely it is that a particular user would want to purchase the book.

To represent the problem, the system designer must decide exactly what features will be used to represent each item that is for sale. If the system involves more than one type of item, then this becomes significantly more complicated; however, we will focus on the sale of books in our example.

User interface: As mentioned earlier, although it is not the focus of this thesis, a good user interface is often crucial for mixed-initiative systems. In this particular example, however, it is not an important detail; a simple text-based interface will

suffice.

Utilities: The system designer must also decide on how to represent the utility of purchasing a specific book for a particular user. For this example, suppose that the utility function is as follows. Consider the situation in which a book has been purchased for a user and the user has had a chance to read and rate the book. Looking back on the purchase, the utility should depend on the user's eventual rating of the book and on the price that was paid for the book. Also, we shall assume that the utility of a purchase is zero if the user already owned the book (although this might not be the case if the user is a collector of rare editions, for example).

Such a utility function could take on many forms, but for the purpose of our example, the utility of purchasing a book b for a person p will be calculated as:

$$U(b,p) = R(b,p) - pen_p(b, cost(b))$$

where $R(b,p)$ is the system's guess at the rating from 0 to 100 that person p would assign to book b and where $pen_p(b, cost(b))$ is a function assigning a penalty to paying price $cost(b)$ for book b for person p. For example, if $max(b,p)$ is the maximum that a person is willing to spend for book b, then one possible penalty function would be:

$$pen_p(b, cost(b)) = \begin{cases} R(b,p), & \text{if } cost(b) > max(b,p) \\ e^{\frac{cost(b,p) \times \ln(R(b,p))}{max(b,p)}} - 1 & \text{otherwise} \end{cases}$$

The idea is for this penalty function to be exponential in the cost of the book and to range from a penalty of 0 for acquiring the book for free to a penalty of $R(b,p) - 1$ for paying the maximum acceptable price.

Of course, prior to purchasing a book, the system will probably not know the user's rating of that book. However, it will use its domain knowledge to *predict* the user's rating in order to get an *estimated* utility of the purchase.

The utility of *not* purchasing a book will depend on the person's "indifference threshold" $IT(p)$. The idea is that, if the utility of buying a book is $IT(p)$, then the utility of not buying the book has the same value. The user is indifferent to buying the book or not buying it. Using this threshold, the utility of not purchasing a book is defined as

$$U(\neg b, p) = min(100, max(0, 2IT(p) - U(b,p)))$$

For example, suppose a person's indifference threshold is 75. If the utility of buying a book is 90, then the utility of not buying it is 60. If the utility of buying the book is 70, the utility of not buying it is 80. If the utility of buying it is 50 or lower, the utility of not buying it is 100.

Task criticality: In this example, the task criticality should be quite low. We are not talking about major purchases. If we choose incorrectly, the user has not spent huge sums anyway. (If we are talking about a domain where the user has the ability to return any unsatisfactory purchases, then the task criticality should

be even lower – although, of course, it is still desirable to avoid causing the user to have to do this.) For our examples, we will set the task criticality of the book-purchasing domain to 20 on a scale from 0 to 100.

User knowledge: As outlined in Section 3.8, the system will make use of information about the knowledge of specific users, about stereotypes of users, and about the general user population to determine the likelihood of a user knowing the answer to a particular type of question.

However, in this particular domain, almost every question will be about the user's own preferences. For a domain of this type, it is safe to assume that the probability of a user knowing about her own preferences is 1. The P_{UK} variable introduced in our model is designed more for domains where the system's knowledge base will sometimes lack certain *factual* information that users might be able to provide. It *is* possible that such questions could come up in the book-purchasing domain – for instance, the system might not know how to classify a particular book into one of the designated genres. In such a situation, it is possible that the user might be able to help to classify the book. Again, it is the responsibility of the system designer using our model to think of the different possible types of questions that could come up in a given domain, and to determine the appropriate role for information about user knowledge.

Another possible use for the P_{UK} variable in this domain is to think of it not as a measure of the likelihood of the user having the required *knowledge* to answer a question, but as the likelihood that the user will *care* about the answer. It is entirely

feasible that certain questions might come up where a user's natural response is simply "I don't care." If this is truly what the user thinks, then it would make sense for this to be an acceptable response that the system is able to deal with. On the other hand, it is not a particularly useful response for the system and should be treated essentially the same way as an "I don't know" response. If such a response is considered to be likely, then the value of asking this question should diminish, just as it does when the system believes that the user is unlikely to know something.

Note that a user response of "I don't care" is not the same as that user being unwilling to interact. Generally speaking, a user might be very willing to communicate with the system; however, there might be certain aspects of a domain that are just not important to that user.[1]

World beliefs: The system's database must be initialized with information about available books, specifying values for as many of the features as possible. Knowledge can also be provided to the system about the general preferences of users, in terms of genres, authors and specific books. This will be very helpful in early interactions with users, when there is very little information available in the form of explicit user ratings. For example, the data shown in Table 4.1 might be provided to the system. This table indicates, for instance, that the average rating expected for mystery books will be 70 across all users. However, for users in specific stereotypical groups, the average expected rating for mystery books might be higher

[1]For this example, we are not discussing the solution in terms of the alternative, information-theoretic model. Note that with this approach, the extent to which a user "doesn't care" should influence the values assigned for the importance of questions.

or lower.

	Average expected rating				
		Stereotypes			
Genre	**All users**	**1**	**2**	$\cdots$	**n**
Mystery	70	90	65	$\cdots$	40
Biography	60	65	90	$\cdots$	50
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Romance	40	20	20	$\cdots$	90
Author	**All users**	**1**	**2**	$\cdots$	**n**
John Grisham	80	95	60	$\cdots$	30
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Danielle Steel	45	15	30	$\cdots$	95

Table 4.1: Initial beliefs about preferences of average users

Time and time criticality: In our examples, we will assume a highly competitive environment. If decisions are not made fairly quickly about a book purchase, then the book will likely be bought rapidly by another customer. More specifically, we will define the following role for time.

Each available book will have a starting time T_s and an ending time T_e associated with it. The utility of purchasing the book for the user after the ending time should be 0. Prior to the end of the sale, there should be a cost associated with delaying the purchase of a book, since long delays will increase the possibility of the item being bought by another user. If t is the length of a delay (in our case, we're interested in delays associated with communicating with the user), and if T_c is the current time, then the cost associated with that delay is defined as

$$c_t = \begin{cases} U(b,p), & \text{if } t > T_e - T_c \\ \frac{t}{T_e - T_s} \times U(b,p), & otherwise \end{cases}$$

Bother cost: In this domain, the cost of bothering a user is quite low. The questions that will be asked of the user are not difficult, not very time-consuming and, in moderation, not likely to frustrate most users. Presumably, any user that is making use of such a system is interested in having the system purchase books on her behalf and would be open to the idea of being asked an occasional question. We will use the bother cost function presented in Chapter 3, using a willingness value of $w = 9$ out of 10. The bother so far is

$$BSF = \sum_I c(I)\beta^{t(I)})$$

where the different values of I represent all the interactions that have taken place so far in this session, where $t(I)$ is the number of time steps that have elapsed since interaction I, and where $c(I)$ is an estimate of the cognitive cost or difficulty of the question asked in interaction I. The actual cost of bothering is

$$b = INIT + \frac{1 - \alpha^{BSF}}{1 - \alpha}$$

where the formulas for $INIT$ and α are as suggested in Chapter 3: $INIT = 10 - w = 10 - 9 = 1$ and $\alpha = 1.26 - 0.05w = 1.26 - 0.05(9) = 0.81$. We assume that all interactions I will have a cognitive cost of $c(I) = 1$ associated with them.

Other costs; weights on costs: We will assume that only time and bother are considered as cost factors. Suppose the weights are initially set to $w_t = 0.3$ and $w_b = 0.2$ to reflect the designer's beliefs about the relative importance of the time cost and the bother cost.

Stereotypes: At this point, the system designer should also establish a set of stereotypes into which users will be roughly classified. For our examples, suppose that a stereotype has been created to correspond to each of the book genres in our system (mystery, biography, romance, entertainment, sports, cooking, business, technical, science-fiction). Users will be placed in a class if they read books from the corresponding genre and very little else. In a full-fledged system, it would not be possible to categorize all users very effectively with this system, and in fact it should be possible for users to belong to several different stereotypes at once, so real-life stereotypical classes would be more detailed than this. However, this simple categorization will be sufficient for demonstrating the model.

4.1.2 First interaction with a user

Once the system has been designed and implemented, it will begin to interact with users. At first, every user will be new to the system. Whenever a new user is encountered, the following variables must be determined.

Willingness to interact: The process of determining how willing the user is to interact with the system can be as simple or as complicated as desired by the designer. Probably the simplest approach is to ask the user to rate their willingness to interact, either on a quantitative scale (e.g., 0-10) or in qualitative terms that could then be converted to a numerical scale.

For our examples, we will be looking at three different users. User 1 is very willing to interact with the system (rating of 9), user 2 is not very willing at all (rating of 1), and user 3 is in between (rating of 5).

Stereotypes: We will now assign each new user to one of our user stereotypes. In the book-purchasing domain, this could be done by asking the user to rate her preference for each genre of book that might be purchased. This constitutes a fairly small set of questions and would probably not be seen as unreasonable by very many users. The results of this survey would then be used to find the stereotype that most closely matches the user's tendencies. Again, in our simple examples, we are only defining stereotypes by the one book genre that interests a reader the most.

For our examples, we will place user 1 in the mystery stereotype, user 2 in the sports group and user 3 in the romance class.

Preference information: The system will also want to know a few other details of the user's preference profile. For example, a rough model of the user's spending profile would be desirable. If a book is in condition x and the original sale price was y, what is the maximum that she would be willing to spend for this

book? What is the maximum amount of money that the user would ever be willing to have the system spend on a book without explicit consultation?

The table below shows how much each user is willing to have spent on their behalf, as well as the maximum they are willing to spend on books in each possible condition (where y is the original sale price of a book).

User	**Max. $ to spend on a book without explicit permission**	**Condition**				
		Poor	Fair	Good	VG	Excellent
1	20	$0.3y$	$0.5y$	$0.7y$	$0.8y$	$0.9y$
2	50	0	0	0	$0.4y$	$0.8y$
3	50	$0.5y$	$0.7y$	$0.8y$	$0.9y$	y

4.1.3 Sample interactions with a user

As specified in Section 3.7, a system that follows our model should be set up so that its knowledge can be updated as it interacts with different users. This section describes a few sample interactions with users, to illustrate how such updates would take place. Again, the types of updates that will be necessary are the following:

Update beliefs about the world

Update user knowledge information

Update willingness information

Maintain dialogue history

In the following examples, S represents the system and U represents the user. The

comments in brackets show the actions of the system involving updating its stored information.

Example 1: The system is considering automatically purchasing a book for the user, but because of prior knowledge about the types of books the user owns, it believes that there is some probability that the user already owns the book. It is trying to determine whether to ask the user if he owns the book or to go ahead and make a decision without asking.

Suppose we have the following expected utilities for the different possible outcomes. For example, if the system buys the book when the user does already own it, the expected utility is 0.

$U(\text{buy} \mid \text{own}) = 0$

$U(\text{buy} \mid \neg\,\text{own}) = 90$

$U(\neg\,\text{buy} \mid \text{own}) = 100$

$U(\neg\,\text{buy} \mid \neg\,\text{own}) = 60$

If the system does not ask, the two possible actions available are to buy the book or not buy the book. The expected utility of buying the book is $P(\text{own})\ U(\text{buy} \mid \text{own}) + P(\neg\,\text{own})\ U(\text{buy} \mid \neg\,\text{own}) = 0.5\ (0) + 0.5\ (90) = 45$. The expected utility of not buying the book is $P(\text{own})\ U(\neg\,\text{buy} \mid \text{own}) + P(\neg\,\text{own})\ U(\neg\,\text{buy} \mid \neg\,\text{own}) = 0.5\ (100) + 0.5\ (60) = 80$. The best course of action then, without acquiring any further information, is to forgo buying the book. $EU_{\neg ask} = 80$.

If the system *does* ask the user, there are two possible responses that could be

obtained. With probability 0.5, the user will say that he *does* already own the book. In this case, the system would not buy the book, with utility 100. With probability 0.5, the user will say that he does not own the book. In this case, the system would go ahead and buy the book, with utility 90. Therefore, the overall expected utility of the system's course of action after asking the question is $EU_{ask} = 0.5(100) + 0.5(90) = 95$.

The benefits of asking are $EU_{ask} - EU_{\neg ask} = 95 - 80 = 15$, but we must compare these benefits to the expected costs.

Suppose that the book is available for only one hour and that the expected time to receive an answer from the user is 5 minutes. Based on the time penalty function described earlier, the time cost would work out to $\frac{5}{60}(90) = 7.5$. Suppose also that the user has not yet been bothered, and has indicated a willingness value of $w = 9$. The bother so far is then $BSF = 0$ and the cost of bothering the user is $b = INIT + \frac{1-\alpha^{BSF}}{1-\alpha} = 1 + \frac{1-0.81^0}{1-0.81} = 1$. If the weights are $w_t = 0.8$ and $w_b = 0.3$, then the total costs are $0.8(7.5) + 0.3(1) = 6.3$. The benefits exceed the costs, and so the system would decide to ask the question. The full interaction might look like this:

S: Do you already own "The Game" by Ken Dryden?

U: Yes.

< S records that U owns book #2304291.>

< S automatically asks a follow-up question. >

S: What is your rating of this book (0-100)?

U: 75.

< S records the new rating and adjusts the user's rating for books of the same genre as book #2304291.>

Also, for each question that the system asks, it records an instance of interaction in its dialogue history with the user. This information captures what questions were asked of the user and at what times.

[Time: 2003-05-19 14:15:23 own(user 003712, book 2304291)]

[Time: 2003-05-19 14:15:30 rate(user 003712, book 2304291)]

Example 2: As a variation on the first example, consider a case where the system is in fact quite sure that the user does not already own the book. Suppose the probability that the user owns the book is only 0.05.

In this case, the system's best choice if it does *not* ask the user is to go ahead and buy the book, with an expected utility of 85.5. If it does ask the user, the expected utility is 90.5. The benefits of asking therefore have a value of 5; this does not exceed the costs, and so the system would decide to forgo asking the user and would automatically purchase the book on the user's behalf.

4.2 Scheduling examples

In this section, we will present a few examples from the domain of sports scheduling. In this domain, a system is given the task of scheduling games that are to take place between a set of teams, along with a number of hard and soft constraints. Hard constraints are not allowed to be violated. Any candidate schedule that fails to satisfy all hard constraints is unacceptable. Soft constraints are only preferences; if a schedule fails to satisfy a soft constraint, then it is not perfect, but might still be acceptable. In fact, it might be the best possible schedule if no perfect schedule can exist under the given constraints.

Suppose that a schedule is evaluated as follows. A perfect schedule – one that satisfies all hard and soft constraints – is given a score of 100. If any hard constraints are violated, the score is 0. For each time that soft constraint i is violated, p_i points are subtracted from the perfect score, where p_i is the penalty associated with constraint i.

4.2.1 Example 1.

In this example, the system has been provided with a list of six teams (A-F) and has been asked to schedule a round-robin schedule, meaning that each team must play one game against each of the other five teams. There are thus 15 games to schedule in total. Suppose we have exactly 15 time slots available – 7:30, 8:30 and 9:30 on each of five days – and that only one game can be played in each time slot. The system is also given the following hard constraints.

Hard constraints:

H1. No team can play twice on the same day.

H2. A cannot play at 9:30.

H3. B cannot play at 7:30.

H4. C cannot play on Day 1.

There is no way to satisfy all of these constraints, because of H1 and H4. Team C must play five games, but cannot play on Day 1, as stated in H4. Therefore, C must play five games in four days, but H1 states that no team can play more than one game on any given day.

Let us first assume that $EU_{\neg ask} = 0$, since there is no way for the system to provide an acceptable schedule.

There are a few possible things that the system might learn by asking the user for help: (1) It might be possible to add a new time slot that fixes the problem; (2) H1 might remain a hard constraint, but might be relaxed to say that no team should play two games *in a row* on the same day; (3) H1 might be turned into a soft constraint with some associated penalty.

The question is: how can we assign a value to EU_{ask}, the expected utility of the system's course of action if it does ask the user for help? (The question in this case would consist of the system describing the problem to the user and asking the user to edit the problem description.)

One approach is to determine this value by reasoning about which of the three possibilities the user is likely to choose, and also about what penalty would likely be associated with the new soft constraint mentioned in the third option.

Suppose that we can estimate that there is a 20% chance that the user will add a new time slot that will solve everything, a 30% chance that he will modify H1 so

that no team can play twice in a row, and a 50% chance that he will turn H1 into a soft constraint with a penalty of 10. In the first two cases, it is possible to come up with a perfect schedule (score = 100), while in the third case, the soft constraint would have to be violated for two teams, giving a score of 80. Now, EU_{ask} would be $(0.2)(100) + (0.3)(100) + (0.5)(80) = 90$.

Assume that we are certain that the user will have the required knowledge to help. Also, suppose that the time cost and bother cost have been set to values of 10 and 8, respectively, and that the weights are $w_t = w_b = 0.3$. Then:

$$Benefits = 90 - 0 = \mathbf{90}$$

$$Costs = 0.3(10) + 0.3(8) = \mathbf{5.4}$$

$Benefits > Costs$, so the system will ask the user.

Until now we have assumed that $EU_{\neg ask} = 0$. Yet the system should be able to reason about some default course of action. Suppose the agent's default action is to design the best schedule it can, *assuming* that H1 has been turned into a soft constraint with some penalty p_1. It can come up with a schedule in which two teams play twice in one day, resulting in a score of $1 - 2p_1$. With an expected value of 10 for such a penalty, the utility of the expected outcome if we *do not ask* would be $1 - 2(10) = 80$.[2] Since EU_{ask} was computed earlier to be 90, the benefits (90 - 80 = 10) would still outweigh the costs (5.4), and the user would still be consulted. Let us now consider a scenario where the system may decide *not* to interact.

[2]Alternatively, the system might have a probability distribution on its belief about p_1 – e.g., 10 with 60% probability, 20 with 30% probability and 30 with 10% probability. This would mean that the expected value of p_1 would be 15, and the utility of the expected outcome would be $1 - 2(15) = 70$.

4.2.2 Example 2.

Round-robin schedule; teams A-F; 15 available time slots (7:30, 8:30, 9:30; Days 1-5).

Hard constraints:

H1. No team can play 2 games in a row.

H2. A cannot play at 8:30.

H3. B cannot play at 7:30.

H4. A cannot play at 9:30 on Day 4.

H5. B cannot play at 9:30 on Days 1–3.

Soft constraints:

S1. A should play F on Day 5.

S2. No team should play twice on the same day.

No perfect schedule exists. By the hard constraints, A must play against B on Day 5. By S1, A should also play against F on Day 5, which would violate S2. The question is: should we violate S1 and move A–F to another day, or violate S2 and have A (and at least one other team) play twice in a day?

Suppose that the user has specified a penalty of 10 for violating S2, but we do not know what the penalty is for S1. However, from similar scheduling scenarios from the past, the system believes that S1 will have a penalty of 10 with probability 0.6 and a penalty of 50 with probability 0.4. The system has come up with two potential solutions, one that violates S1 and one that violates S2 twice (two teams have one day each on which they play twice).

Schedule 1					
Time	1	2	3	4	5
7:30	A-E	A-D	A-C	**A-F**	C-D
8:30	B-C	B-E	B-F	C-E	E-F
9:30	D-F	C-F	D-E	B-D	A-B

Schedule 2					
Time	1	2	3	4	5
7:30	A-E	A-D	A-C	**C-D**	**A-F**
8:30	B-C	B-E	B-F	E-F	C-E
9:30	D-F	C-F	D-E	**B-D**	**A-B**

The utility of the second schedule is known to be 80. The *expected* utility of the first is $0.6(90) + 0.4(50) = 74$, so the system might make its best guess and choose the second one. The utility of this is $EU_{\neg ask} = 80$.

The alternative is to ask the user to specify the correct penalty for violating constraint S1. Suppose the system *does* ask. With probability 0.6, it expects that the user will give the low penalty (10) for violating S1. In this case, it can come up with a schedule with utility 90. With probability 0.4, it expects the higher penalty for violating S1, in which case the *other* schedule (the one which violates S2 twice and has a score of 80) would be chosen in the end. Therefore, if we ask the user, we would expect a 60% chance of choosing a schedule with utility 90 and a 40% chance of a schedule with utility 80. The utility of the expected outcome is therefore $(0.6)(90) + (0.4)(80) = 86$.

Again, assuming the same costs as above, and assuming that we are sure that

the user will have the needed knowledge, the computations are as follows.

$$Benefits = 86 - 80 = \mathbf{6}$$

$$Costs = 0.3(10) + 0.3(8) = \mathbf{5.4}$$

$Benefits > Costs$, so the system will ask the user.

However, if the value of P_{UK} were even slightly lower, the benefits would no longer exceed the costs. For example, if $P_{UK} = 0.8$, the benefits would drop to 4.8, and the system would decide not to ask.

4.3 Translation

In this section, we will discuss another potential application area: interactive translation. In the examples discussed in this section, a system has been asked to translate a document for the user and must make decisions on when to interact with that user to obtain more information. The examples in this section will focus on the heuristic model presented in Section 3.6.

Example 1: In this first example, the system is translating a document from English to Japanese. The user is the author of the original English document, but does not know Japanese at all.

Consider the situation in which the system is unsure of how to translate a particular sentence in the user's source document into Japanese. There are at least two possible approaches here.

- One approach is to try to disambiguate the original sentence in English: to provide the user with possible interpretations of the original sentence and to ask what the intended meaning was. Figure 4.2 shows an example.

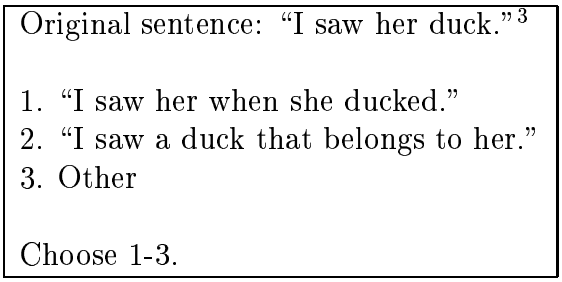


Figure 4.2: Clarifying an ambiguous sentence

In this example, the number of possible answers is 3. If the system believes that interpretations 1 and 2 are equally likely and that there is only a 10% chance that the user will choose to provide a third interpretation, then its information-theoretic uncertainty about the answer is $I(0.45, 0.45, 0.10) = 1.37$. Suppose the importance of the question is quite low (25 out of 100). We can assume that the user who wrote the original sentence is very likely ($P_{UK} = 0.99$) to know how to choose from a list of possible intended meanings. The constant κ has been learned to be $\kappa = 0.27$.

The expected benefits of asking are then:

$$\begin{aligned} Benefits &= \kappa \times P_{UK} \times Imp \times Uncertainty \\ &= 0.27 \times 0.99 \times 25 \times 1.38 \\ &= 9.22 \end{aligned}$$

[3]This example of an ambiguous sentence was taken from
http://www.ohiou.edu/ linguist/soemarmo/l270/Exercises/ambigs/ambigs.htm.

The expected costs are a weighted linear combination of all costs that have been identified for the given domain. In this case, suppose the time cost for asking the user for help is quite low (25). The task criticality is moderate (50) because the user is trying to translate a summary of her research interests, to be published on her web page. The user has indicated a very high willingness to be bothered (w=9) and so the bother cost function is:

$$\begin{aligned} b &= INIT + \tfrac{1-\alpha^{BSF}}{1-\alpha} \\ &= (10-9) + \tfrac{1-(1.26-0.05\times 9)^{BSF}}{1-(1.26-0.05\times 9)} \\ &= 1 + \tfrac{1-0.81^{BSF}}{1-0.81} \end{aligned}$$

If the user has been bothered twice so far, at 3 and 7 time steps in the past, then the bother so far is

$$\begin{aligned} BSF &= \sum_I c(I)\alpha^{t(I)} \\ &= (1)0.95^3 + (1)0.95^7 \\ &= 1.56 \end{aligned}$$

The overall cost of bothering the user is then $1 + \frac{1-0.81^{1.56}}{1-0.81} = 2.47$. Finally, if the weights on time, bother and task criticality are $w_t = 0.2$, $w_b = 0.5$ and $w_c = 0.1$, then the total costs of interacting are

$$\begin{aligned} Costs &= w_t t + w_b b - w_c Taskcrit \\ &= (0.2)(25) + 0.5(2.47) - (0.1)(50) \\ &= 1.24 \end{aligned}$$

The benefits exceed the costs, and so the system will ask the user for help.

- A second approach to this same situation would be to provide several possible *Japanese* translations and to ask the user which one would be preferable. Of course, if the system knows that the user does not know Japanese, then this seems like an unwise choice. However, we will demonstrate here how our heuristic model would deal with this situation.

  Everything would be identical to the previous case except that P_{UK} would be zero. In this case, the costs would remain the same, but the benefits would become

  $$Benefits = \kappa \times P_{UK} \times Imp \times Uncertainty$$
  $$= 0.27 \times 0 \times 25 \times 1.38 = 0$$

  Now, the costs are greater than the benefits and the system would correctly decide not to interact with the user.

Example 2: As a variation on the first example, consider the second situation described above (asking the user to choose the most appropriate Japanese translation), but now with a user who *does* have some knowledge of Japanese. Suppose P_{UK} is set to 0.7 in this case. Again, all of the computations would remain identical except for the benefits calculation, which would become

$$Benefits = \kappa \times P_{UK} \times Imp \times Uncertainty$$
$$= 0.27 \times 0.7 \times 25 \times 1.38 = 6.52$$

Benefits exceed costs, and the system would choose to interact.

4.4 Military planning

A very common application area for research on mixed-initiative interaction has been military planning (e.g., (Cox and Veloso, 1997)). Although we have not investigated this domain in detail, we present two very brief examples here to demonstrate the importance of task criticality and time criticality in our model.

Example 1. Suppose that, in a particular military scenario, a system has determined that the benefits of interacting with the user will be quite small ($EU_{ask} = 85$ vs. $EU_{\neg ask} = 82$).

However, the system's evaluation of the costs of interaction reveals that the criticality of the task is so high that it overrides the contribution of the time cost and bother cost, to the point that the total overall costs end up being negative.

$t = 25$

$b = 5$

$Taskcrit = 100$

$w_t = 0.2$

$w_b = 0.5$

$w_c = 0.1$

$Costs = w_t t + w_b b - w_c Taskcrit = 0.2(25) + 0.5(5) - 0.1(100) = -2.5.$

Although the idea of negative costs might seem somewhat counter-intuitive, recall that the costs in our model represent the *additional* costs involved in interacting with the user, as opposed to having the system act without further consultation.

In this example, the negative costs represent the fact that it is in fact *less costly* to interact, because of the potential cost of a critical error if we do not take the opportunity to ask the user for assistance. In the end, the decision in this case would be to interact with the user; even though the benefits appear to be quite small, the high criticality of the task leads the system to conclude that the benefits outweigh the costs.

Example 2. Now consider a similar military scenario, but where the system has the additional constraint that the problem is *highly* time-critical. This could be represented in the system's knowledge by assigning a utility of 0 to any course of action that will cause the system to complete its task after a specific time T. If any interaction with the user is expected to take longer than T, then any sequence of actions involving asking the user for help will be guaranteed to have a utility of 0.

The result is that the benefits of interacting would actually turn out to be highly negative. For instance, if the system can achieve an expected utility of $EU_{\neg ask} = 60$ by choosing an action on its own without consulting the user, but any interaction with the user is guaranteed to yield $EU_{ask} = 0$, the result is that $Benefits = EU_{ask} - EU_{\neg ask} = 0 - 60 = -60$.

In this case, the benefits will almost certainly be exceeded by the costs, and the system would correctly determine that it should do the best it can, as quickly as it can, without consulting the user.

Chapter 5

Experimentation and Results

It has been suggested (Chin and Crosby, 2002) that, historically, user modeling research has been lacking when it comes to the defence of models through empirical evaluation. In this chapter, we provide a modest but still significant attempt to defend the model presented in this thesis through experimentation.

The experiments in this chapter consist of simulations of user-system interactions in a generic problem-solving setting. For future work, we will extend this work by measuring the performance of our systems on real-world tasks, working with real users. However, the simulations presented in this chapter represent a solid first step in defending the usefulness of our model.

In this chapter, we demonstrate that improved performance *can* be achieved by modeling the user's knowledge about different topics and by estimating the degree to which a user will be bothered by an interaction. We also provide a validation of the information-theoretic model of Section 3.6 by showing that it can perform quite well despite not doing a complete analysis of all possible future events and

interactions.

5.1 Defense of Factors

5.1.1 User Knowledge

The results presented in this section serve two purposes: (1) to illustrate the fact that systems that model the probability of the user being knowledgeable will outperform systems that do not, and (2) to demonstrate that, even if the system's estimate of this probability is not perfectly accurate, it is still advantageous to model this factor.

In this experiment, different agents using a dynamic programming approach[1] were presented with a generic task to perform. The agents differed only in their modeling of the user's knowledge; this will be discussed further after the problem description.

Problem description

The task involves selecting one of three possible actions (labelled 1, 2, and 3). The reward associated with performing each of these three actions depends on the actual values of three independent variables (A, B, and C). All of the variables are binary (either 0 or 1). At the beginning of the task, the agent is presented with the reward

[1]Recall that, in Chapter 3, it was discussed that dynamic programming is the simplest way of extending our "single-decision" model to deal with sequential decision problems. Dynamic programming allows for the expected utility of every state to be computed simply by working backwards from the known utilities of final states. It is a reasonable approach for situations, such as the ones described in this chapter, in which the agent's action sequences are finite. When infinite sequences are possible, the Markov decision process approach becomes necessary.

function, and so it is aware of exactly how the rewards depend on A, B, and C. However, it does not have any information about the actual values of A, B, and C. Initially, it believes that each variable is equally likely to have the value 0 or 1. The only way to obtain more accurate information is by asking the user.

There are two costs associated with interacting with the user: it uses up time and it inconveniences the user. The costs associated with each of these factors are subtracted from the agent's eventual reward according to a function that is known to the agent. The actual performance of the agent is determined according to the formula below, where $R(s, a)$ is the reward for performing action a in state s, t is the elapsed time, b is the cost of bothering the user, and w_t and w_b are the weights indicating the relative importance of these costs.

$$\text{Performance} = R(s, a) - w_t \times t - w_b \times b$$

To complicate the situation, there is no guarantee that the user would, in fact, have the knowledge required to answer a question when asked. The user's ability to answer a question is determined by a parameter that is set at the beginning of each trial. There is also a possibility that, even if the user does provide a response, there is a chance that the user's answer will turn out to be incorrect.[2]

At each time step, the agent has to decide whether to ask the user about variable A, B, or C, or to commit to choosing one of the three actions (1, 2, or 3). Once it has committed to an action, the task is complete and it receives its reward. Intuitively,

[2]In the experiments described in this section, the probability of the user answering incorrectly is set to 0.25.

the rational way for such an agent to behave is to make the choice with the highest expected utility, given its current knowledge. Basically, it should continue to ask the user questions until there is no question for which the expected benefit of acquiring that knowledge would outweigh the costs of bothering the user and wasting time. The benefit is determined by considering how much it would improve its expected performance on the task if it were able to reduce its uncertainty about A, B, or C. Again, since this is a small problem with a finite number of time steps, the expected utilities were computed using a dynamic programming approach.

Experimental set-up

For each trial, the values for the reward function were chosen as follows.

1. Each of the three variables A, B and C was first assigned a random weight so that the three weights summed to 1. For example, variable A might have weight 0.32, variable B 0.55, and C 0.13.

2. For each possible value of each variable, a score was then assigned for each possible action 1, 2 and 3. These scores were then scaled so that the best action would receive a score of 100 and the worst would receive a score of 0. For example, if variable A is equal to 0, then the score associated with performing actions 1, 2 and 3 might be 43, 100 and 0.

3. Rewards were then computed for each possible action under every possible *combination* of values for the three variables A, B and C. This was done using the weights from part 1 and the scores from part 2. An example appears in

Table 5.1.

weight(A) = 0.32	weight(B) = 0.55	weight(C) = 0.13
score(A=0,action=1) = 43	score(B=0,action=1) = 100	score(C=0,action=1) = 0
score(A=0,action=2) = 100	score(B=0,action=2) = 0	score(C=0,action=2) = 87
score(A=0,action=3) = 0	score(B=0,action=3) = 23	score(C=0,action=3) = 100
score(A=1,action=1) = 0	score(B=1,action=1) = 12	score(C=1,action=1) = 100
score(A=1,action=2) = 59	score(B=1,action=2) = 100	score(C=1,action=2) = 0
score(A=1,action=3) = 100	score(B=1,action=3) = 0	score(C=1,action=3) = 37
reward(A=0,B=0,C=0, action 1) = $0.32 \times 43 + 0.55 \times 100 + 0.13 \times 0 = 68.76$		
reward(A=0,B=0,C=0, action 2) = $0.32 \times 100 + 0.55 \times 0 + 0.13 \times 87 = 43.31$		
$\vdots$		

Table 5.1: Computing rewards for experimental trials

The complete reward function was then made available to the agent. The actual values for A, B, and C were chosen randomly, but this information was *not* made available to the agent. The (simulated) user, however, might know each of these pieces of information with some probability. The weights, w_t and w_b, associated with the time and bother penalties were also set at the beginning of each trial. The cost of time was simply a linear function: for each time step that elapsed before the completion of the task, a cost of w_t was subtracted from the total reward. The bother function was as described in Section 3.4.3, with the willingness value set to 8.

Results

In this section, we will look closely at a typical set of trials. In this example, the actual probability of the user providing an answer to each question was set at 0.7.

The time and bother weights were set to 5 and 1, respectively.

Eleven different agents were presented with exactly the same situations. There were 1000 trials in total. The first seven agents modeled the probability of the user knowing the answer to a question. Only agent 4 had the correct value for P_{UK} (0.7); the values of P_{UK} used by agents 1-7 were, respectively, 0, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0.

Figures 5.1 and 5.2 show the average performance by each of agents 1-7, where the performance is calculated by taking the reward earned by an agent and subtracting the time and bother costs incurred. As expected, the agent that models P_{UK} *correctly* has the best performance. Also, we can see that, as we approach the actual P_{UK} value from either direction, the agents' performance improves. This illustrates the second point that was mentioned at the beginning of this section: that even if the system's estimate of P_{UK} is not perfectly accurate, it is still advantageous to model this factor.

The worst results were achieved by Agent 1, which assumes that the user will never know the answer to any question and by Agent 2 ($P_{UK} = 0.5$). Agent 7, which assumes that the user will always know the answer, also performs relatively poorly. It is this last agent that is the most relevant comparison point for our model. If a system assumes that the user will always know the answer to a question, then essentially it is not modelling P_{UK} at all. The agent that is able to model P_{UK} correctly outperforms this system.

Figure 5.3 shows the agents' performance in terms of the percentage of trials in which the best possible choice was made, given the actual values of variables A, B

Agent #	Assumed P_{UK}	Average performance
1	0.0	68.99
2	0.5	70.50
3	0.6	70.80
4	0.7	71.41
5	0.8	71.33
6	0.9	71.07
7	1.0	70.87

Figure 5.1: Performance of agents assuming different P_{UK} values

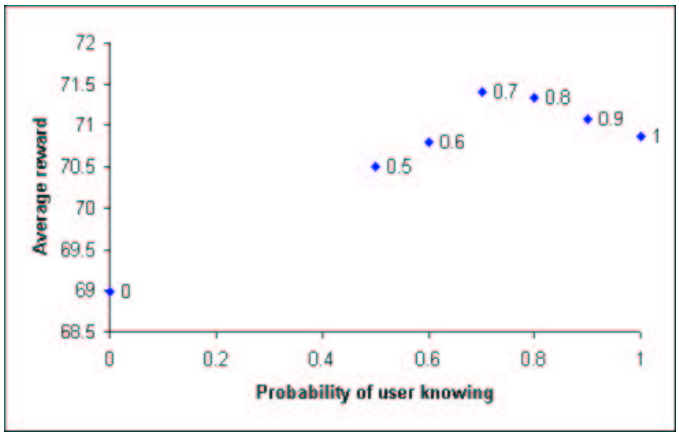


Figure 5.2: Graph of performance of agents assuming different P_{UK} values

and C. Not surprisingly, agent 7 had the best results; because it believed that the user was certain to know the answer, it would ask questions more often. Because of this, it gathered more information from the user and made correct decisions more often. However, this gain in decision accuracy was not enough to counteract the additional cost incurred, as shown in Figure 5.1 where the costs of communication were also incorporated into the overall measure of success.

Agents 8-11 are used to illustrate the value of using our model as opposed to problem-solving approaches that do not use intelligent techniques. Agent 8 does not

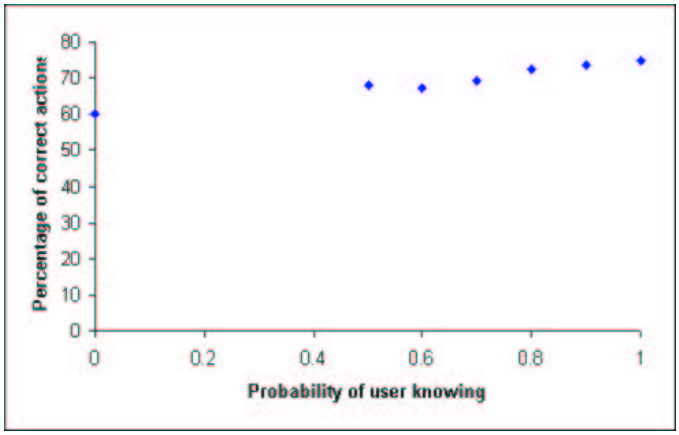


Figure 5.3: Percentage of correct action choices by agents assuming different P_{UK} values

consider P_{UK} at all, nor does it use any of the information it is given about how the rewards depend on the values of variables A, B and C. It simply chooses an action at random on the first time step. Agent 9 chooses a random action (including possibly asking about A, asking about B or asking about C) at every time step. It will not ask a question that it has already asked, but otherwise its decisions about actions are completely random. Agent 10, meanwhile, tries to gather as much as possible without regard to the cost of asking. In every trial, it will ask about variables A, B and C, and then choose the action with the highest reward. Finally, Agent 11 chooses one question (asking about A, B or C) randomly and asks this question. It then chooses the action that has the highest expected utility, given its knowledge after asking the question.

The average performances achieved by agents 8-11 were 52.65, 42.95 and 48.24 and 61.17, respectively. These are well below the results obtained by the systematic approaches. Of these four agents, the most meaningful basis for comparison is Agent

10. It can be viewed as a sensible, deliberative agent, but one that does no user modeling. The fact that our agents outperform this agent gives evidence that our approach is beneficial.

Note that Agent 1 can also be viewed as an agent that does no user modeling. In this case, however, instead of asking every possible question (as Agent 10 does), Agent 1 will *never* decide to ask the user a question. It will always choose the action that has the best expected utility with the information currently available to the agent. Again, Agent 4 (the agent that models P_{UK} correctly) outperforms Agent 1, demonstrating that agents that do not employ any user modeling at all are indeed inferior to our agents.

The fact that Agent 4 achieved the best results of all the agents is a positive result. However, the margin by which this *correct* agent has the best average performance is narrower than expected. This is likely due in part to the reward functions being chosen randomly, rather than reflecting the patterns that would be exhibited by reward functions in a more realistic domain. For future work, more elaborate experiments must be performed, using actual reward functions elicited from real users.

Specific examples

In this section, we present two specific examples from our experimental trials of how agents can go wrong if they incorrectly model the probability of the user knowing the answer to questions.

Example 1. In the first example, the agent's reward function was as shown in Table 5.2. The actual values of A, B and C in this trial were 1, 1 and 0, respectively.

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	47.84	49.91	51.25
0	0	1	0.00	99.91	11.32
0	1	0	96.84	16.69	51.00
0	1	1	49.00	66.69	11.08
1	0	0	47.84	50.00	50.83
1	0	1	0.00	100.00	10.91
1	1	0	96.84	16.78	50.59
1	1	1	49.00	66.78	10.66

Table 5.2: Reward function for example 1

The agent that correctly modeled the probability of the user knowing ($P_{UK} = 0.7$) chose to ask the user about variable C. Upon receiving the correct answer ($C = 0$), it chose to perform action #1 and achieved a performance of 89.84 (reward of 96.84, minus 7 for time and bother costs).

The agent that assumed that the user had a P_{UK} of 0.0 did not perform as well. It chose to perform action #2 without any interaction with the user, due to the perception that the user would be unlikely to know the answer. Although action 2 looked like the best possible action, it turned out to be a poor choice because of the actual values of A, B and C, and the performance achieved by the agent was only 16.78.

Example 2. In this example, the agent's actual reward function was as shown in Table 5.3. The actual values for the variables A, B and C in this trial were 1, 0 and 0, respectively.

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	34.69	90.00	28.68
0	0	1	31.30	90.00	37.03
0	1	0	68.69	56.00	12.77
0	1	1	65.30	56.00	21.12
1	0	0	66.00	79.47	28.68
1	0	1	62.61	79.47	37.03
1	1	0	100.00	45.47	12.77
1	1	1	96.61	45.47	21.12

Table 5.3: Reward function for example 2

The agent that correctly modeled the probability of the user knowing ($P_{UK} = 0.7$) chose to perform action #2 without any interaction with the user. It achieved a performance of 79.47.

The agent that assumed that the user had a very high probability of knowing the answer ($P_{UK} = 1.0$) chose to ask the user about variable B. After asking this question, it actually received the wrong answer ($B = 1$) from the user and ended up choosing action #1 and achieving a low performance of 59.00 (66.00, minus 7 for time and bother costs).

5.1.2 Varying P_{UK} for each question

In another set of trials, the experimental set-up was changed so that the simulated user did not have exactly the same P_{UK} value for every question (e.g., 0.7 in the example described above). Instead, at the beginning of each trial, a value was chosen randomly for *each* question from the possible values 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0. This is closer to a real-life situation, where a user might be quite likely to

be able to answer some questions, but not others.

Four different agents were tested in this situation: one that correctly estimates the P_{UK} value for each question, one that averages the actual P_{UK} values for the three questions and uses this average as an estimate of P_{UK} across all of the questions, one that uses $P_{UK} = 0$ for all questions, and one that uses $P_{UK} = 1$ for all questions.

The average performances achieved by these agents were 73.13, 72.97, 71.03 and 72.54, respectively. Again, the agent that models the real world correctly (Agent 1) achieves the best performance. Agent 2 does almost as well by using an average P_{UK} value across all questions, while Agents 3 and 4 do not perform as well by assuming that the user will never/always be able to answer.

Specific examples

In this section, we present two specific examples from our experimental trials of how agents can go wrong if they incorrectly model the probability of the user knowing the answer to questions.

Example 1. In this example, the agent's reward function was as shown in Table 5.4. The actual values of A, B and C in this trial were 1, 1 and 0, respectively. The actual probabilities of the user knowing the answer to questions A, B and C were 0.6, 0.9 and 0.9, respectively.

The agent that correctly modeled the probabilities of the user knowing (0.6, 0.9, 0.9 for questions A, B and C) chose to ask the user about variable B. Upon receiving the correct answer ($B = 1$), it chose to perform action #1 and achieved a performance

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	0.00	44.00	96.52
0	0	1	0.00	42.82	96.52
0	1	0	81.00	75.89	15.52
0	1	1	81.00	74.70	15.52
1	0	0	17.00	27.00	96.24
1	0	1	17.00	25.82	96.24
1	1	0	98.00	58.89	15.24
1	1	1	98.00	57.70	15.24

Table 5.4: Reward function for example 1

of 91.00 (reward of 98.00, minus 7 for time and bother costs).

The agent that assumed that the user had a P_{UK} of 0.0 did not perform as well. It chose to perform action #3 without any interaction with the user, due to the perception that the user would be unlikely to know the answer. Although action 3 looked like the best possible action, it turned out to be a poor choice because of the actual values of A, B and C, and the performance achieved by the agent was only 15.24.

Example 2. In the second example, the agent's reward function was as shown in Table 5.5. The actual values of A, B and C in this trial were 1, 0 and 0, respectively. The actual probabilities of the user knowing the answer to questions A, B and C were 0.7, 0.6 and 0.8, respectively.

The agent that correctly modeled the probability of the user knowing (0.7, 0.6, 0.8 for questions A, B and C) chose to perform action #1 without any interaction with the user. It achieved a performance of 100.00.

Both the agent that assumed that the user had a very high probability of know-

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	80.00	28.92	32.80
0	0	1	68.92	43.92	20.00
0	1	0	27.79	75.09	32.80
0	1	1	16.71	90.09	20.00
1	0	0	100.00	18.84	18.93
1	0	1	88.92	33.84	6.13
1	1	0	47.79	65.00	18.93
1	1	1	36.71	80.00	6.13

Table 5.5: Reward function for example 2

ing the answer ($P_{UK} = 1.0$) and the agent that used the average P_{UK} of 0.7 chose to ask the user about variable B. After asking this question, they actually received the wrong answer ($B = 1$) from the user and ended up choosing action #2 and achieving a low performance of 11.84 (18.84, minus 7 for time and bother costs).

5.1.3 Bother Factor

Similar experiments were performed to measure the usefulness of modeling the user's willingness factor as part of computing the cost of bothering the user. Again, a number of agents were tested on sets of 1000 trials. Each agent had a different assumption about the willingness level of the user.

For example, tests were run with a simulated user having an actual willingness level of 6, but with different agents assuming willingness levels of 0, 5, 6, 7, 8, 9 and 10. The average performances of these agents are shown in Table 5.6. As with the experiments devoted to modeling user knowledge, the trials demonstrate that better results are obtained as the system's model of the user's willingness level gets

closer and closer to the actual value.

Agent #	**Assumed w**	**Average performance**
1	0	72.32
2	5	73.20
3	6	73.39
4	7	72.64
5	8	72.42
6	9	70.99
7	10	70.91

Table 5.6: Results from bother factor experiments

Specific examples

In this section, we present two specific examples from our experimental trials of how agents can go wrong if they incorrectly model the user's willingness to interact.

Example 1. In the first example, the agent's reward function was as shown in Table 5.7. The actual values of A, B and C in this trial were 1, 0 and 0, respectively.

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	32.40	61.00	39.00
0	0	1	25.33	72.00	34.55
0	1	0	28.99	4.56	100.00
0	1	1	21.92	15.56	95.55
1	0	0	28.87	89.00	11.00
1	0	1	21.80	100.00	6.55
1	1	0	25.46	32.56	72.00
1	1	1	18.39	43.56	67.55

Table 5.7: Reward function for example 1

The agent that correctly modeled the user's willingness level ($w = 6$) chose

to ask the user about variable B. Upon receiving the correct answer ($B = 0$), it chose to perform action #2 and achieved an overall performance of 82.00 (reward of 89.00, minus 7 for time and bother costs).

The agent that assumed that the user had a willingness level of 0 did not perform as well. It chose to perform action #3 without any interaction with the user, due to the high perceived cost of the communication. Although action 3 looked like the best possible action, it turned out to be a poor choice because of the actual values of A, B and C, and the performance of the agent was only 11.00.

Example 2. In this example, the agent's actual reward function was as shown in Table 5.8. The actual values for the variables A, B and C in this trial were 0, 1 and 1, respectively.

Actual values			Action choice / reward		
A	B	C	1	2	3
0	0	0	37.69	79.00	42.16
0	0	1	23.96	79.00	42.16
0	1	0	60.61	27.00	68.46
0	1	1	46.87	27.00	68.46
1	0	0	16.69	81.04	46.69
1	0	1	2.96	81.04	46.69
1	1	0	39.61	29.04	73.00
1	1	1	25.87	29.04	73.00

Table 5.8: Reward function for example 2

The agent that correctly modeled the user's willingness level ($w = 6$) chose to perform action #3 without any interaction with the user. In doing so, it achieved a performance of 68.46.

The agent that assumed that the user had a very high willingness level ($w = 10$)

chose to ask the user about variable B. After asking this question, it actually received the wrong answer ($B = 0$) from the user and ended up choosing action #2 and achieving a low performance of 20.00 (27.00, minus 7 for time and bother costs). Even if it had received the correct answer from the user in this case, however, it would have made the same decision as the $w = 6$ agent (action #3) but achieved a lower performance of 61.46 (reward of 68.46, minus 7 for time and bother costs).

5.2 Validating the information-theoretic model

To demonstrate the value of the information-theoretic model presented in Section 3.6, agents were implemented that did not project into the future at all. They simply used the information-theoretic model to estimate the value of asking each question, using their uncertainty about the possible answers to the question, the perceived importance of the question and the probability of a successful interaction with the user.

Several different information-theoretic agents were implemented, using different values for the constant κ. The results showed that, by properly choosing κ, it is possible for an information-theoretic agent to perform almost as well on this task as the agents using the full-fledged dynamic programming approach, despite the fact that no reasoning is being done about how the user's answer might actually impact the choice of action that the system would make and the possible sequences of states that might be visited as the problem execution continues.

On the same 1000 trials used for the user knowledge experiments, a heuristic agent with κ set to 11 achieved an average performance of 70.51, which is quite

close to the performance of the best agent (average 71.41) and is in fact better than the performance of the agent that assumed $P_{UK} = 0.5$ instead of the correct value of 0.7.

Chapter 6

Discussion

6.1 Extensions to the Model

6.1.1 Understandability

As mentioned in Section 3.3.2, it is possible for our model to take into account the idea of the understandability of a potential question for a user. When the system is deciding whether or not to ask the user a question, it could consider whether or not the user will have enough background contextual information to be able to understand the question and the system's underlying goals for asking it (Fleming and Cohen, 2000).

This can be accomplished by replacing the variable P_{UK} in the model with a new variable that represents the probability that the user both has the knowledge and will understand the question. We make the assumption that the interaction will only be successful if *both* of those conditions hold. Even if the user has the

required knowledge but does not understand, or understands the situation but does not have the required knowledge, he will still be unable to help the system.

In fact, we can generalize this idea even further by talking about a general variable to represent the probability of *successful communication*. If designers of particular systems can think of relevant factors other than the knowledge of the user and the ability of the user to understand the question, then the definition of this variable would simply need to be adjusted, to incorporate those factors.

A very simple formula for this variable would be $P_{UK} \times P_{UU}$, where P_{UU} is simply the probability that the user would be able to understand the question, given the current context and dialogue history. This formula includes an implicit assumption of independence between the user's knowledge and the understandability of the question. In fact, we would expect some connection between these two factors: users who are more likely to understand are probably more likely to be knowledgeable users as well, and vice versa. If understandability is to be incorporated as a full-fledged component of our model, then dealing with this interdependence is an important avenue for future work.

In the case that a system *does* determine that a question is likely to be difficult to understand, one possibility is to initiate a clarification subdialogue with the user. The purpose of this subdialogue would be to familiarize the user with difficult terminology or to bring the user up to date with the current state of the problem solving, in order to make the question more understandable. In our model, the initiation of such a dialogue can simply be treated as another action available to the system. If it is decided that a particular question should not be asked, because

its benefits do not outweigh its costs, then a clarification dialogue should be initiated if the expected benefits of the clarification exceed the expected costs.

6.1.2 Predictability

Another interesting extension is the idea of incorporating the *predictability* of a system's actions into its decision-making procedure.[1] What role should the user's *expectations* play in dictating the best course of action for a system at any given time?

Consider a situation in which the system has determined that the rational action is not to interact with the user at that particular time. However, it believes that the user will *expect* to be asked a question in this situation, based on the way previous dialogues have progressed. How can the decision-making procedure be altered so that this additional factor is taken into account?

If Markov decision processes are being used to model the dialogue, as in Section 3.5, then a small reward could be added for performing actions that have been performed successfully in similar situations in the past. Metrics such as those used by Kozierok (1993) would be used to measure the similarity between the current situation and situations encountered in the past, and to compute a score for each possible action based solely on what was done in the past. This score would then be multiplied by a weight factor determined by the importance that the user has attributed to system predictability. This would yield the additional predictability-based reward that would be associated with performing this action.

[1]The author acknowledges Anthony Jameson (German Research Center for Artificial Intelligence (DFKI)) for suggesting the idea of modeling the predictability of a system's actions.

For the information-theoretic model of Section 3.6, the formula for the value of asking a question would remain the same as it was before, but with an extra term added to the end. This term would be the same small weighted reward described in the previous paragraph.

In either case, the result would be that, in some situations in which the system would have decided *not* to ask the user for help, it would now decide to ask (or vice versa). If the weight attributed to predictability is very low (or zero), then this would rarely (or never) occur. However, if it is decided that the user's expectations of system behaviour are in fact quite important, then such situations would arise more frequently.

6.1.3 User Availability and User Attention

The models developed in this thesis required a user model. Decisions made by the system about whether or not to interact relied on user-specific information such as the user's perceived ability to answer questions, the user's preferences among different possible outcomes, and the user's willingness to interact with the system. Two considerations that could also be incorporated into a decision-making model are user availability and user attention.

User availability

In some cases, the system might have reliable information about whether or not the user is even available to answer a question if needed. This would be possible, for example, if the system were equipped with a camera able to detect the user's

presence in the room, or if the system had information indicating that the user was scheduled to be in a meeting at that time.

Such information could be incorporated into the model as follows. If the system believes that the user is unavailable, then the benefit of interacting should be scaled according to the probability of the user being available. For example, if the system is certain that the user is not available, then the benefit of interacting right now should be zero. Of course, it is possible that the system might be totally incapable of proceeding without the user's help, in which case it will just have to wait until the user returns and then ask for help.

The costs of asking a question should also be adjusted when the user might be unavailable. Suppose the user is believed to be available with probability P_{avail}. If the standard formula for bother cost (from Section 3.4.3) yields a value b, then we should adjust this cost to be $P_{avail}b$. This accounts for the fact that, if the user is in fact unavailable (with probability $1 - P_{avail}$), then there will be no cost associated with bothering the user.

However, the time cost would increase in this situation since the system might end up waiting a very long time for a reply from the user. If the expected time associated with getting a response from an available user is t, then we should adjust the expected time value to $P_{avail}t + (1 - P_{avail})T$, where T is a built-in deadline after which the system will stop waiting for a response from an unavailable user.

User attention

Even if the user is available to answer a question, his attention might be focused elsewhere, causing him to be less willing to interact with this system at a given

moment than he would be in general. Horvitz, Jacobs and Hovel (1999) considered the idea of probabilistically modeling the user's focus of attention and using this information to determine the cost of alerting the user about e-mail. Such an approach would be a useful extension to our work.

The cost of bothering should be higher if the user is currently busy with other important things. If we could somehow estimate the user's current cognitive load, then this could be used as a factor. One possible mechanism for this is simply to tweak the current value for the user's willingness; if the user rated their willingness to interact as a 7 out of 10, but the system knows he is very busy right now, then this value could be lowered temporarily to 6 or 5. The degree of change would depend on the estimated cognitive load (exactly *how* busy/distracted is the user?).

Of course, if no such information is available to the system, one possibility is to leave it up to the user entirely to adjust the willingness level when appropriate.

6.1.4 User Initiative

The focus of this thesis has been on providing a framework for systems to use in deciding when *they* should take the initiative to solicit further information from the user. However, a mixed-initiative system must also be able to deal with the fact that, in many domains, users will want to take control of the dialogue themselves, possibly at unexpected times.

The models in this thesis rely on a view of problem solving in which the current situation is always represented as a *state*. The state representation captures everything about the current situation that is needed to distinguish it from other

possible situations. This includes information about the current status of the problem solving and of the dialogue between the two parties. Any actions initiated by the user, whether they involve advancing the problem solving process or simply communicating with the system, will directly influence the current state of the system. Once the user has taken the initiative to take an action, the system will determine the new state that it finds itself in, and will use this new state to make decisions about further actions and interactions.

One important point that is not the focus of this thesis, but is crucial to the effective design of mixed-initiative systems, is the idea that systems should be designed so that the computer is able to determine the intentions of the user as much as possible. There is thus a delicate balance between empowering the user and ensuring a smooth dialogue. If the system is not equipped with sophisticated natural language understanding capabilities, then the user should be limited in the types of utterances that he can make, perhaps by restricting user initiative to a finite set of menu commands, so that the system can understand without excessive clarification dialogues. However, if the user's options are too limited, then the difficulty of using the system will frustrate users to the point that it will not be considered a worthwhile system to use.

6.1.5 Deeper analysis of questions being asked

In this thesis, we have factored out the consideration of generating appropriate questions for the user (assming that a module will provide this information). A few important research challenges arise, as follows.

Improper questions

If the question-providing module proposes questions that turn out to be irrelevant to the user's task, this will result in more bother and cost to the user. The topic of considering the cost of irrelevant questions is addressed by Raskutti and Zukerman (1997).

In addition, if there are difficulties with the actual plan recognition that determines the user's task, therefore influencing the questions to be selected, this will also result in costly interaction. We have not explored the case of faulty plan recognition, but this topic is discussed by Wu (1991), in the context of a decision procedure for generating dialogue.

For future work, we could assume that the question generating module may be faulty, incorporating probabilities that irrelevant or improper questions may be generated and adjusting the calculations accordingly.

Follow-up questions

Certain questions to be asked of the user will necessarily lead to further, follow-up questions, all as one dialogue between the system and user.

Although much more complex modeling would be possible, we view such follow-up questions in this thesis simply as an extension of the initial question. If such follow-up questions are believed to be likely, then the expected cost of the interaction will reflect this belief. For instance, if the expected cost of a primary interaction is calculated to be 10 and if a follow-up question with an estimated cost of 5 is expected to be necessary with probability 0.7, then the overall expected cost of the

original question would be $10+0.7(5) = 13.5$. We assume that such follow-up questions will be asked automatically if they are deemed necessary, and that no further reasoning will be done to decide on whether or not to ask the follow-up question. Similary, the expected benefits of an interaction are meant to reflect the benefits that would be achieved if a question (and any necessary follow-up questions) were asked of the user.

To effectively calculate the long-term benefits and costs of interacting, it is ideal to reflect all the ensuing work throughout the follow-up dialogue in the calculations. For future work, we could tease apart these calculations more precisely and examine more examples that explicitly require follow-up questions.

Raskutti and Zukerman (1994; 1997) provide further discussion on the issue of follow-up questions.

6.1.6 Heuristic functions for system uncertainty

In Section 3.6, an information-theoretic approach to designing mixed-initiative systems was developed. However, it was pointed out that a system that lacks the ability to predict the different possible answers to a question and the expected probabilities of each of those answers will not be able to use such an approach. Such a system will be forced to use a heuristic measure to capture its general uncertainty about its domain knowledge.

In this subsection, we propose that such a heuristic should take into account the system's experience with working in the domain, the system's experience working with this particular user, as well as any information that the system has about its

performance on similar tasks in the past.

To measure a system's overall experience in the domain, we use the following formula:

$$EXPERIENCE = w_{user}h_{user} + w_{class}h_{class} + w_{total}h_{total}$$

Each of the h terms represents a heuristic evaluation of the system's historical performance in interacting with, respectively, the specific user involved, the stereotypical class of users to which the user belongs, and all users in the domain. The details of the computation of these heuristic values are presented below. Again, the weights in the formula are determined by the system designer, according to the estimated importance of each of the component terms.

To compute each of the h terms in the above formula, we measure the system's past performance on previous tasks using the formula below. This formula is computed three times, once for the specific user, once for the stereotypical user class, and once for all users.

In each case, the situations s_i in the formula are the k situations from the system's history that are deemed to be the most similar to the current situation s, according to a distance metric $distance(s, s_i)$ such as the one proposed by Kozierok (1993). The variable k, the number of past situations examined by the system, is specified by the designer of the system. $Success(s_i)$ is a measure of how well the system performed when it encountered situation s_i.

$$h = \sum_{s_i} \frac{Success(s_i)}{k + distance(s, s_i)}$$

If, in a particular system, it is impossible to measure the similarity between two situations, then the $distance(s, s_i)$ should be replaced in the formula by a constant k' chosen by the system designer. This formula will, of course, be less informative than the original one. However, because of the success rating in the numerator, a system will be more confident in its domain knowledge if it knows that it has performed well in the past than if it has performed poorly.

Similarly, the success rating in the numerator can be replaced by a constant k'' if no information is available about the system's past performance. Again, this formula will be less informative than the original one. However, it will do the best with the information it has available: because of the distance metric in the denominator, a system that *has* seen very similar situations in the past will be more confident in its domain knowledge than a system that has not encountered similar situations.

Once we have completed the heuristic evaluation of the system's experience, we interpret this as a measure of the system's confidence in its domain knowledge. The formulas presented in Section 3.6 relied on a measure of the system's *uncertainty* in its abilities. We compute this uncertainty simply by taking $1 - EXPERIENCE$. A system that has a great deal of experience should have a low uncertainty value, and vice versa.

6.1.7 Multiple users

A very interesting possibility is to extend the models to cases where more than one human user is available as a potential collaborator for the system. Such a

system would model the expertise and willingness to interact for *each* of these users. In any situation in which user assistance is a possibility, the system would then perform a costs-benefits analysis for interacting with each of these possible sources of information, in order to determine whom to ask.

After receiving new information from a user, the system would then reevaluate its situation, determining the benefits and costs of additional interactions. It might turn out to be advantageous to ask *another* user the same question, in order to reinforce the system's belief that the information acquired from the first user was accurate.

Another interesting twist presented by the case of multiple users is the need to balance the preferences of different users in situations in which the goals and plans of the various users involved might conflict. For example, consider a system being used to design a teaching schedule for a large academic department. All other things being equal, the preferences and constraints of a permanent faculty member might hold more weight than those of a temporary instructor when trying to resolve conflicts.

6.1.8 A Qualitative Approach

In this section, we provide a more conservative approach to making decisions about interaction with users. Up to this point, we have used probabilistic reasoning to estimate whether or not the benefits of interacting are likely to outweigh the costs. The procedure presented in this section is a qualitative recasting of this decision problem. Some system designers might consider this a more appropriate approach

for certain application domains – or in general, depending on their opinions regarding logical and probabilistic reasoning.

Also, for certain applications, there may be insufficient data on which to make reasonable estimates of the values required by the quantitative model. Our qualitative decision process, which is dependent on evaluating certain binary conditions about the user, is a liberal strategy proposing that the system initiate interaction unless it has reason to believe that interaction with this user would *not* be successful.

Included in our decision process are some factors employed in the quantitative calculation: whether the user has the knowledge required and whether the user can understand the interaction. Since there is no independent calculation of expected benefits and costs, we incorporate two important conditions which are tied to these factors: whether the user is willing to interact and how important is the task to be performed by the system. We also allow for clarification, in cases where the system feels it is important to interact but the user would not be able to understand without further explanation. The overall algorithm is shown in Figure 6.1.

Although there are no numeric values to reason with during this deliberation process, information about the user still needs to be used. The system might determine that the user does not have the knowledge it is seeking during the interaction, by stereotyping the user and labelling certain classes of facts as unknown to this user class. Alternatively, the system can decide that it does believe the user has the knowledge it seeks, either from a stereotyping of the user or by examining the past history of interactions with this user, to conclude that this user knows facts

```
If not (System Believes not (User-Knows-About(p)) then
  If not (System Believes not (User-is-Willing-to-Interact-about(p)) then
    If System Believes (User-Can-Understand-Interaction-about(p))then
         System Asks User (p)
    Else
         If System Believes (User-can-be-made-to-understand(p)) then
            System initiates learning phase for User
            System Asks User (p)
         Else
            System Acts without Asking User
  Else
    If System Believes (Very-Important(p)) then
         /* interact even if User unwilling */
         System Asks User (p)
    Else
         System Acts without Asking User
Else
  System Acts Without Asking User
```

Figure 6.1: Algorithm for qualitative model

of this type. Stereotypes and past history could also be used in the evaluation of the system's beliefs about the user's willingness and ability to understand. As for determining that the task at hand is "very important," this would be done on the basis of the system's view of the problem solving task.

The kind of reasoning required to determine the values for conditions in the qualitative model could also be employed in coming up with strategies for calculating the numeric values required in the quantitative formulation. For example, a user who has been unwilling to interact a certain number of times in the past could be assessed as having a certain bother cost in the cost calculation.

6.1.9 When do we reason about interaction?

An important consideration in designing mixed-initiative systems that reason about the benefits and costs of interaction is the question of *when*, and how often, to perform that reasoning. In most artificial intelligence applications, the system's behaviour over the course of a problem-solving session can be viewed as a sequence of steps. The system performs one action after another as it works its way towards completing the task. In many systems, particularly the mixed-initiative systems that are the focus of this thesis, the system's actions will be intertwined with actions performed by the user as they cooperate on performing the task.

In the MDP model presented in Section 3.5, the system does not have to reason about interaction at run-time. Such a system will have at its disposal an optimal policy that has been computed ahead of time. This policy will provide the system with information about the best action to take in any state that might arise over the course of a session.

With the dynamic decision network approach mentioned in Section 3.5.4, time is broken down into discrete time steps. At each such time step, the system can determine the best decision to take by considering each possible action it might take and using Bayesian inference techniques to determine which action has the highest expected utility.

For the heuristic model of Section 3.6, we propose that the system take a similar approach. It should reason about interaction before every action it takes. Since the calculations in these models are fairly straightforward, the amount of time required for these computations will be minor – and certainly much smaller than the time

required for Bayesian inference.

Furthermore, in many cases, the reasoning of the system will not even involve resorting to the quantitative reasoning provided in our models. If a system knows of a step-by-step plan to achieve a particular subgoal within a domain and knows that it has all the information it will need as it executes that plan, it can determine that there is no need to consider interacting with a user until that subgoal has been completed. Similarly, in other situations, a system might not be able to proceed at all without getting more information from a user. In this case, there is no need to reason about the benefits and costs of asking the question; the system should just go ahead and ask the question immediately.

6.1.10 Reasoning about Reasoning

An important point that has not yet been raised is the fact that reasoning about interaction can itself be costly. In some cases, a system might spend a significant amount of time and computational effort weighing the pros and cons of interacting with the user. It will certainly be true in some situations that the system would have been better off simply to make a decision based on very rough guidelines rather than undertaking full-fledged reasoning about the interaction decision.

An extreme example of this was mentioned earlier. In cases where a system simply cannot make any progress in its problem solving without garnering additional information from the user, then it must certainly try to obtain this information. Utility-based reasoning about whether the interaction's benefits will outweigh its costs is pointless; it *will* be beneficial – unless the system is absolutely certain that

the user will be unable to help.

If a system designer believes that this is almost always the case in his application of interest, then the qualitative model presented in Section 6.1.8 might prove to be superior to the quantitative models developed in Chapter 3. If this is not *always* the case in a domain, but if it is expected to be a concern sometimes, then a hybrid technique might be required. An important future research question is how to effectively develop such a hybrid system.

6.2 Research Contributions

The work described in this thesis has contributions to make to several different research areas. Those contributions will be discussed under separate headings in this section.

6.2.1 Mixed-initiative systems

Past research efforts in mixed-initiative interaction have been largely independent of one another; system designers have primarily come up with their own mixed-initiative solutions to suit their domain of interest. What is needed is a principled, domain-independent framework for designing mixed-initiative systems.

One of the important components of such a framework is a decision procedure to be used by mixed-initiative systems in determining *whether or not* to interact with a user in a given situation. Such systems will often be faced with situations in which it is believed that the user might be helpful, but in which there is also some cost associated with such an interaction.

We have developed a model in which such decisions can be made by weighing the perceived benefits and costs of the interaction. We have elucidated the factors that must be used in determining those benefits and costs. Such decisions must be made not only on the basis of the system's current problem-solving state, but also according to the history of the system's dialogue with the user and according to the system's model of the user's knowledge, preferences and abilities.

As mixed-initiative interaction becomes more prominent in an increasing variety of application areas, including robotics (Kortenkamp et al., 1997), intelligent tutoring (Carberry, 1997; Lester et al., 1997; Shah and Evens, 1997) and planning (Burstein and McDermott, 1996; Allen, 1994), it is important for developers to avoid the pitfalls of designing ad hoc systems without a solid, principled, theoretical set of guidelines. The research described in this thesis lays the groundwork for the development of such application-independent guidelines.

The model we have developed is general and does not rely on assumptions about any specific application area. Our work uses existing techniques from decision theory, but focuses on specific factors relevant to system-user interaction that should be incorporated into decision-theoretic reasoning. We have provided initial suggestions for how certain factors could be implemented in our model, but yet the framework remains flexible enough to accommodate the application-specific needs of system designers.

6.2.2 User modeling

We have described how to make use of a user model in decisions about interaction by mixed-initiative systems. In particular, in our model, we have specified the role that should be played by the user's knowledge, by the user's preferences among possible outcomes in a domain, and by the user's willingness to interact with the system.

The idea of using a user model to determine *whether* a system should interact with a user, as opposed to *how* to interact, or how to modify the text generated by an automated system, is an interesting question for the user modeling community and a novel contribution of this work.

In Section 3.8, a method was described for combining information about the knowledge of a *specific* user, of users of a certain *stereotype*, and of all users *in general*. Techniques for incorporating both general and specific user modeling information are lacking in the literature. The framework proposed in this thesis represents a solid step toward establishing methods for weighing the relative contributions of these different types of user model, and for designing systems that *adjust* the importance of specific and general information as the system gains experience.

Adaptable systems are becoming more common in our society, particularly in the form of adaptable hypermedia systems on the World Wide Web (Brusilovsky and Maybury, 2002); user models play a crucial role in such systems. The research described in this thesis has a strong connection to the user modeling community, and is therefore highly relevant in the development of intelligent systems in the near future.

6.2.3 Discourse

Although it has not been a primary focus of the work, the research described in this thesis will be of interest to the discourse research community. We are addressing a discourse problem: one in which a collaborative problem-solving session between a system and a user is treated as a dialogue between the two parties. This dialogue is one in which either party might take control of directing the task at any time.

Although we do not have a discourse model, in the way that a system like Collagen (Rich and Sidner, 1998) has – with a focus stack, natural language generation, and so on – we do incorporate a very basic discourse model into our overall model, by keeping a record of certain aspects of the dialogue thus far. This includes a record of the number of times that the system has requested help from the user, the times at which those interactions took place, and the estimated cognitive effort that these interactions required of the user. Based on this information about previous instances of interrupting the user, we compute the estimated cost associated with bothering the user in the current situation.

Our view of initiative is primarily one that fits under Chu-Carroll and Brown's definition of dialogue initiative (Chu-Carroll and Brown, 1997). Although a general mixed-initiative system will have to adopt a broader view, our focus has been on situations in which the system is attempting to decide whether or not it should take the dialogue initiative to ask a question of the user. In essence, a system that takes this type of dialogue initiative is in fact actively *relinquishing* the task initiative, by suggesting that the user should take more control of the task by guiding the system.

6.2.4 Decision Theory

Our work continues a recent trend of using decision-theoretic techniques to model dialogue (e.g., (Haddawy and Hanks, 1998; Paek and Horvitz, 1999b; Murray and Lehn, 2000)). We propose that a Markov decision process model of dialogue will be appropriate for dialogue situations that satisfy certain criteria (see Section 3.5). However, we also identify certain limitations of these techniques for modeling dialogues. In situations in which these limitations prove to be problematic, we suggest an alternative, information-theoretic approach to modeling the benefits and costs of interacting with a user in a cooperative problem-solving session.

6.2.5 Multi-Agent Systems

The main message to be taken away from this thesis, with respect to multi-agent systems, is that we are calling for more synergy between the multi-agent research community and the mixed-initiative research community. In our research, we have hopefully helped to elucidate why that synergy is needed.

The two areas are in fact very similar in many ways. Work such as that of Gmytrasiewicz and Durfee (2001) and Xuan, Lesser and Zilberstein (2001) has a lot to offer to researchers attempting to design systems that make intelligent decisions about interacting with human users. Essentially, whether the other participant(s) in a dialogue are humans or other artificial agents, many of the same techniques can be used. The benefits and costs of initiating an interaction should be analyzed, and communication should be initiated only if the benefits exceed the costs.

We have also suggested one major difference between communication in multi-

agent systems and in mixed-initiative systems: the cost associated with bothering the user. In Section 3.4.3, we discussed the crucial role that this bother factor should play in a mixed-initiative system, and we provided some ideas on how such a factor should be computed.

Closely related to the area of multi-agent systems is the research field of adjustable autonomy (Hexmoor, Falcone and Castelfranchi, 2003). Systems with adjustable autonomy are able to behave differently in different situations, in terms of how proactive they are in accomplishing a task or in terms of the amount of decision-making control or authority they adopt. Although adjustable autonomy is often studied in the context of a community of intelligent agents, our work focuses on the case of only one intelligent system working in tandem with one human user. Although, in some sense, this situation is simpler because there are only two decision-makers involved, it is made more complex by the psychological aspects of human-computer interaction, including the bother factor discussed in the previous paragraph.

In this chapter, we have briefly described some of the possible extensions to our work that could lead to valuable future projects in mixed-initiative system design. We have also identified several important contributions of our work to research in mixed-initiative systems, user modeling, discourse, decision theory and multi-agent systems.

In Chapter 7, some of the related work in these fields will be described, and compared and contrasted with our research.

Chapter 7

Related Work

This chapter includes descriptions of several examples of related work, drawn from the research areas of mixed-initiative systems, decision-theoretic reasoning, spoken dialogue agents, multi-agent systems, clarification dialogues, collaborative interface agents, intelligent tutoring, and user modeling. Each section includes some discussion about the relationship between our work and the research being described.

7.1 Horvitz et al.

Horvitz (1999) discusses the debate among AI researchers between the need to design better automated services (interface agents) and the need to improve direct manipulation capabilities for users. His view is that the idea of mixed-initiative interaction provides a nice middle ground between these two extremes. In particular, he identifies factors that he believes are essential for the "effective integration of automated services with direct manipulation interfaces."

Horvitz (1999) also looks at the idea of systems making decisions about interaction based on the expected utility of actions. In this particular paper, the application domain is that of meeting scheduling. As a new e-mail message is being read by the user, the agent tries to decide whether or not it should assist the user by beginning the process of scheduling a meeting in the user's electronic appointment book. If so, it brings up the appointment book in a new window, filling in information as specifically as it can. It is also able to identify conflicts with existing appointments and propose possible alternatives.

Here, the decisions being made by the system are relatively straightforward. Should it (a) do nothing at all, (b) begin the scheduling process, or (c) initiate a dialogue with the user to ask about her goals?

The decisions are made based on the simple idea of expected utility, according to the following information that is available to the system:

- the probability that the user has the goal of scheduling a meeting
- the six utilities associated with the system {performing the scheduling action, initiating a dialogue, doing nothing} when the user {does, does not} have the goal of scheduling a meeting

In this domain, the probability of the user wishing to schedule a meeting is computed by looking at patterns in the text of the message and by considering previous habits of the user.

Default utilities are provided by the system designers, but they can be altered by the user if desired.

Using this information, along with ideas from basic decision theory, the system can compute two threshold probabilities $p^*_{\neg A,D}$ and $p^*_{D,A}$.[1] Using these thresholds, the system can immediately decide what to do in any new situation. If the probability p of the user wanting to schedule a meeting is below $p^*_{\neg A,D}$, then the agent will do nothing. If $p > p^*_{D,A}$, the system will invoke the automated service. If p is between the two thresholds, the system will ask the user about their goals.

This particular paper deals with a different type of problem from what we are considering in this thesis. Horvitz (1999) is looking at situations where the system is trying to decide whether or not it should invoke an automated service to help the user. If the system did not do so, it would be the user's responsibility to manually invoke the service if there were indeed anything to be done. In our work, we have been considering situations where the system has already been asked to work on some type of problem for the user. Our decisions about interaction have been focused mostly on questions of whether or not the system could benefit from additional help from the user.

However, Paek and Horvitz (1999b) do consider cases that are somewhat closer to our work. In this paper, the *Bayesian Receptionist* is described. It is a system that deals with tasks that are commonly encountered by receptionists at the Microsoft corporate campus. It is the system's responsibility to try to infer the user's goals and to fulfill the request. In one example shown in that paper, the system is uncertain about its understanding of the user's utterance and, after some reasoning about the costs and benefits of different types of questions, decides that it is best

[1] These two variables represent the threshold probabilities between total inaction and initiating dialogue $(\neg A, D)$, and between initiating dialogue and taking an action autonomously (D, A).

to ask the user a question about his goals.

The ideas used by Horvitz are very similar to ours. The actions of the system are dictated by the expected utilities of the possible courses of action. In Horvitz's work, however, the availability of the different probabilities and utilities is more realistic. In this specific instance, a system might be able to reason about the likelihood of a particular message being one that involves a potential meeting to be scheduled. Also, it is reasonable that someone might be able to come up with utilities for the six possible outcomes. For example, how irritating would it be for a user if the system were to pop up a scheduling window when he had no intention of scheduling a meeting?

However, in a general setting, there are far too many utilities to be specified and the probabilities of the user's goals are harder to predict. In our work, we are specifying a framework that can be used for any general application in which a system might decide to interact with the user.

One observation about the relationship between our work and Horvitz's is that our model makes the computation of utilities somewhat more explicit. In Horvitz's work, costs such as time and bother are simply incorporated into the utility values. For example, the utility of the system invoking the automated service when, in fact, the user did not want that done is simply set to be low if the expectation is that this user will be greatly bothered by such an outcome. We have identified the factors that might go into an assessment of such a utility value, so that designers or users can use a systematic approach to assigning those numbers. While Horvitz mentions that he is considering costs and benefits of different actions (Horvitz, 1999), we do

not see that explicitly. We only see utility values that, in theory, incorporate those costs and benefits.

Furthermore, our work addresses the possibility of applications where utility values would not be readily available. In particular, the model presented in Section 3.6 discusses an information-theoretic approach to reasoning about interaction without projecting ahead to possible future outcomes and their expected utilities.

7.1.1 Focus of Attention

Horvitz, Jacobs and Hovel (1999) focus more on the idea of when to *present* information *to* a user, as opposed to our goal of deciding when to *request* information *from* a user. They try to infer the expected criticality of an e-mail message (based on its content) and "balance the context-sensitive costs of deferring alerts with the cost of interruption." They look at different costs of interruption for different types of alerts and for the different possible tasks that the user might be focused on. They provide formulas for the following ideas:

- expected cost of interruption
- expected cost of deferring alerts (cost of delayed action)
- expected value of transmitting an alert

The discussion about costs of delayed action includes a few comments about dividing e-mail messages into different "criticality classes." The cost of delayed review of a message depends on its criticality.

Although we have not really considered the idea of *alerting* the user about information in this thesis, this would be an important part of the larger picture of how mixed-initiative systems should work. Furthermore, the formalism used by Horvitz, Jacobs and Hovel (1999) for alerting the user would probably also apply to the case of when to ask the user. In particular, as discussed in Section 6.1.3, the idea of using information about the user's focus of attention is a possibility for an extension to our model presented in Chapter 3.

7.2 Litman, Walker et al.

The Spoken Dialogue Agents group at AT&T has produced some research that is closely related to our work. PARADISE (PARAdigm for DIalogue System Evaluation) is "a general framework for evaluating spoken dialogue agents" (Walker et al., 1997b). This framework provides a means by which the performance of different agents can be compared. In PARADISE, the idea of an agent's overall *performance* is "modeled as a weighted function of a task-based success measure and dialogue-based cost measures, where weights are computed by correlating user satisfaction with performance" (Walker et al., 1997b). The underlying assumption is that the ultimate goal of such a system is to maximize user satisfaction, and that this objective can be broken down into two sub-objectives: maximizing task success and minimizing the costs associated with inefficient dialogues.

Task success is measured by using the Kappa coefficient (Carletta, 1996; Siegel and Castellan, 1988). Many different cost measures have been identified by the AT&T group, including the number of system and user turns, the number of help

requests and the total elapsed time of the dialogue. By performing regression analysis on a set of data from sample dialogues, they are able to identify which factors are statistically significant in predicting user satisfaction and to determine the relative contribution of each of these relevant factors.

The PARADISE framework has been shown to be very useful for evaluating different versions of the same spoken dialogue system, in order to determine the overall dialogue strategy that appears to be best for a given domain. Some of the AT&T experiments have involved using PARADISE to compare mixed-initiative versus system-initiative dialogue strategies (Walker et al., 1997a), to compare literal versus cooperative responses (Litman, Pan and Walker, 1998), and to investigate the value of a tutorial dialogue to familiarize users with a system before using it (Kamm, Litman and Walker, 1998).

Although there are similarities between the goals of this research and of ours – most notably, the aim to produce dialogues that maximize task success while minimizing dialogue costs – our approaches are significantly different. The AT&T group has looked at techniques for choosing an optimal *overall dialogue strategy* for systems. We, on the other hand, are focused on designing systems that can make decisions about the value of interaction at some step in the *middle* of a problem-solving session. Rather than acting according to a pre-defined dialogue strategy, our agents are able to evaluate situations as they arise and to make appropriate decisions "on the fly."

More recent AT&T research has moved towards the design of systems that can adapt their dialogue strategies during the course of the dialogue: first by allowing

the user to manually adjust the system's strategy during problematic sessions (Litman and Pan, 1999) and then by allowing the system to use learning techniques to anticipate speech recognition problems and to adjust the dialogue strategy accordingly (Litman and Pan, 2000). However, the overall approach is still to have the system behaving according to one of a fixed set of dialogue strategies, rather than treating each individual situation separately.

7.3 Gmytrasiewicz and Durfee

Gmytrasiewicz and Durfee have looked at the problem of rational communication in multi-agent environments (Gmytrasiewicz and Durfee, 2001). The decisions being made by their agents are very similar to those being discussed in this thesis, in that they involve decisions about the value of communication with another agent. Their approach also involves the use of expected utilities to guide the system in its decisions about communication. As in our work, they are looking at different possible actions that could be taken by the agent and attempting to choose the one that will have the maximum benefit. Like us, they deal with decisions about interaction just like decisions about task-related actions.

They look at several different types of communication: messages informing other agents about certain aspects of the world, messages about the "speaker" agent's own intentions, acknowledgment messages, questions, and commands. Their discussion of *questions* is particularly relevant to our research. However, the multi-agent setting is quite different from the mixed-initiative scenario in this regard. In Gmytrasiewicz and Durfee's work, the agents are assumed to be self-interested, whereas

we make the assumption that the human user is likely to help the system as much as possible, since it is often the human user's own goals that the system is trying to fulfill. Gmytrasiewicz and Durfee provide some interesting discussion about the value of representing questions as declarations of ignorance in multi-agent environments.

Another interesting aspect of their research is their use of the Recursive Modeling Method (Gmytrasiewicz and Durfee, 1995) as a representation for the agent's reasoning. This method involves representing not only the agent's beliefs about the world, but its beliefs about other agents in the system, about those other agents' beliefs, and so on. Although it would be an interesting idea for future research to apply this recursive modeling representation to systems using our reasoning model, this has not been in our focus thus far.

The primary difference between their work and ours is the fact that we are dealing with systems where the other agent is assumed to be a human user. As a result, our concern with issues such as the user's probability of being a helpful collaborator and the cost associated with bothering the user are not primary concerns in Gmytrasiewicz and Durfee's work. There are, however, some interesting parallels. In their research, for example, they consider the possibility of communicative acts being unsuccessful due to unreliable communication channels or misunderstandings on the part of the other agent; these ideas are strongly related to our initial discussion about the understandability of system utterances in Section 6.1.1.

Although Gmytrasiewicz and Durfee do consider that there is a cost to communication, these costs are just presented in their work as given values, with no discussion of where cost information would come from. One of the main contribu-

tions of our work is to specify how communication costs, particularly for interaction with human users, should be computed.

One final difference is that their work considers the value of communication in a "myopic" way: only the immediate effects of interactions are considered in computing the utilities; there is no concern for how an interaction might affect the long-term success of the overall communication. In our work, we extend this idea by providing models for incorporating the long-term benefits and costs of interactions. This is addressed in our model by projecting ahead, looking at *sequences* of actions and interactions that might be taking place in the future, and considering the utility of the eventual solution that could be reached after each possible sequence of decisions.[2]

7.4 Sullivan et al.

Sullivan et al. (2000) apply utility-based reasoning to the following multi-agent problem. There is a team of agents and a set of tasks that have been distributed among the various agents in the group. When an agent is presented with an "outside offer" (a potentially rewarding job that is not part of its commitment to the team), should it default on its task(s) within the team framework in order to accept the outside offer?

- One or more agents from the group might be able to perform the defaulting agent's tasks, but those tasks might also go undone.

[2]Note that our treatment of long-term benefits and costs still factors out the topic of a possibly faulty question generating module. This is discussed further in Section 6.1.5.

- The team incurs a cost (divided equally among the agents) whenever an agent defaults on a task.
- Future decisions about the assignment of tasks will be made according to the agent's "score," which is based on its reliability in the past. In other words, an agent that defaults on its assigned tasks now will be assigned less important (and therefore less rewarding) jobs in the future.
- In determining whether to accept an outside offer, each agent determines the utility of the various options. This calculation can include three factors:
  - current income - takes into account the income from the outside offer and the cost of defaulting
  - future expected income - estimate of income from future weeks, according to the likely effect of defaulting on the agent's "score"
  - brownie points - gained or lost according to whether an agent does or does not choose to default on its tasks; considers the values of the offered task and of the defaulted task. This is only used internally by an agent as a measure of social consciousness; it does not actually affect future task assignments.

The utility of defaulting is computed as indicated below, where TEI is the total estimated income (a combination of current income and future expected income), where BP refers to brownie points, and where the weights sum to 1.

$$U_{def} = TEIweight \times normTEI_{def} + BPweight \times normBP_{def}$$

A similar calculation is then done for *not* defaulting. Agents default when $U_{def} > U_{no-def}$.

The basic ideas of their work and ours are the same. Their agents decide to do something (default on their assigned tasks in order to accept a new task) if the expected utility of the new task outweighs the expected utility of the old ones. We decide to do something (interact with the user) if the expected benefits outweigh the expected costs. In fact, the Sullivan approach could very well be applied to our problem of interest. The question of whether or not to default on a task would be replaced by the question of whether or not to ask the user a question. Also, the following interpretations would apply to the terminology used in Sullivan's work.

- The current income could represent an estimate of how well the current task will be done if the agent does (or does not) ask the user for help.
- The future expected income might try to estimate how well the user or system would be able to perform *future* tasks if the system does (or does not) interrupt in this instance. For example, if a system interrupts and expects that what it learns from the user could improve its ability to handle future situations, this would add to the utility of interrupting. Similarly, in a different type of domain, the utility might be increased if there is some expectation that an interaction will improve the *user*'s future ability to perform actions without any further help from the system.
- The brownie point model could correspond to user satisfaction. It would be necessary to incorporate some kind of user feedback here. An agent would

gain brownie points if it chose to interrupt the user and if that interaction proved valuable, or if it chose not to interrupt and if such a situation was later evaluated by the user as being one where interaction was indeed unnecessary. Similarly, an agent would lose brownie points if it incorrectly chose to interrupt or to keep quiet. Unfortunately, this analogy is not likely to work very well. If we really want it to correspond to the brownie point model, then the agent needs to be able evaluate things at the moment at which it is making the decision. If there were some way to compute how "similar" the current situation is to past situations (Kozierok, 1993), then it could use past feedback to evaluate the current situation.

- Another possibility is to use some penalty for bothering the user, instead of a brownie-point system. This could just be some constant value, or could depend on how much the user had been bothered recently, or could try to consider the user's focus of attention. The calculations would then look more like those below, where the bother factor is incorporated only into the utility calculation for interrupting.

$$U_{interrupt} = TEIweight_{interrupt} \times normTEI_{interrupt} - BotherWeight \times normBotherValue$$

$$U_{no-int} = TEIweight_{no-int} \times normTEI_{no-int}$$

Again, the agent would interrupt if $U_{interrupt} > U_{no-int}$.

One important distinction between our work and that of Sullivan et al. is that we must also take into account the *cost* of the interaction, which does

not have an analogue in their work.

In some sense, our problem is a more difficult one to look at in a concrete way. Their agents are dealing with *explicit* rewards that are available for performing certain actions in the domain. In a general mixed-initiative system, the system's goal will be to attempt to perform a task as well as possible and to please the user as much as possible. Often, there will not be a concrete way to evaluate the possible actions, especially not before the task has actually been performed. This is precisely the problem of the availability of utility functions discussed in Section 3.6.

7.5 Xuan et al.

Xuan, Lesser and Zilberstein (2001) also look at the problem of communication decisions for multi-agent systems. They are concerned with the fact that agents are often in uncertain environments and are unable to observe the states of the other agents with which they are working. The agents have complete knowledge of their local state, but not of the global state of the multi-agent system. The framework that Xuan et al. develop for dealing with this situation is similar to a multi-agent Markov decision process (Boutilier, 1999), but one in which individual agents have to make decisions about whether or not to try to observe the global state.

This work is very much related to ours, in that Xuan et al. are interested in developing systems that will treat interaction as a decision-theoretic problem, choosing to initiate communication only if the expected gain outweighs the expected

cost.

Although Xuan et al. address the fact that communication should have a cost associated with it, and although their model allows for this cost to vary according to time and the agent's current state, there is no discussion about *how* this communication cost would be calculated. As stated in the discussion of Gmytrasiewicz and Durfee's work in Section 7.3, we specify *how* the cost of communication should be computed for the case in which the other agent is a specific human user, focusing specifically on the cost associated with bothering this user.

The agent's decisions about whether or not to try to observe the local state of another agent is very similar to our problem of deciding whether or not to ask questions of a user. However, in our case, the questions would have to be much more specific, in order to try to access particular pieces of information about the user's knowledge or preferences. In many situations, it is difficult to ask a human to provide a complete report on their "current state"; there simply would be too much information involved.

7.6 van Beek, Cohen and Schmidt

Cohen et al. (1994) and van Beek et al. (1993) looked at the use of plan recognition in generating appropriate responses from advice-giving systems. In particular, they discussed the problem of when to initiate a clarification dialogue with users. This was done by examining the possible plans that the user might have and by identifying faults with each possible plan. By grouping together plans with the same fault, they were able to make decisions about whether any ambiguities in the system's

view of the user's plan were in fact relevant to the task of generating a cooperative response.

In the work by Cohen et al. (1994), the system would ask the user a clarifying question whenever it was determined that there was ambiguity in the user's plan and that this ambiguity would make a difference to the system's response. Such a clarification dialogue was always assumed to be beneficial. We have extended this work by introducing the idea of *costs* of interaction, and by considering whether or not the benefits outweigh the costs.

We have also extended their work by considering issues such as the perceived knowledge and willingness of the user. In the applications considered by Cohen et al. (1994) and van Beek et al. (1993), users could always be assumed to be knowledgeable about their own goals and preferences. Furthermore, it was assumed (quite naturally) that users of their advice-giving systems would always be willing to be fully collaborative participants in the dialogues. In our more general setting, users might be asked questions about the state of the world – questions that some users might be more or less likely to be able to answer. Also, in many applications, the system might have been enlisted by the user to perform some task on his behalf, with some users being more or less willing to provide assistance to the system. The expected value of asking a clarifying question should depend on how likely we are to get a helpful response and on how bothersome the dialogue will be for the user.

Another extension of the work by Cohen, van Beek and Schmidt is in our introduction of probabilistic information into many aspects of the decision-making situation, particularly in the system's view of its own confidence about what to do

in a problem-solving setting. In Cohen et al.'s work, the algorithms relied on the system being able to determine the user's plan to the point that there was no ambiguity or that any ambiguity would be irrelevant to the system's eventual response. However, building on recent work on decision-theoretic approaches to artificial intelligence (Horvitz, 1988), we allow for the possibility of a system making decisions *without* resolving ambiguities, if it is confident enough in its best guess and if the expected benefit of resolving the ambiguity is sufficiently low.

7.7 Chajewska et al.

Chajewska et al. (2000) have done some interesting work on adaptive utility elicitation. They suggest, as we do, that in many situations, acquiring full utility functions for all users will not be feasible. As a result, intelligent systems must always make decisions with partial utility functions. Although many researchers have looked at the problem of incomplete probabilistic information about the effects of actions, very few have dealt with uncertainty about utilities themselves. Chajewska et al. treat the user's utility value for a given outcome as a "random variable that is drawn from a known distribution" (Chajewska, Koller and Parr, 2000).

The result is an adaptive utility elicitation process that selects the best questions to ask according to their expected relevance (via a value-of-information analysis). The system stops asking questions when the value drops below a given threshold.

This is very similar to our work, except that our reasoning has focused on asking the user questions about the environment, rather than about their utility functions. Extensions to our model incorporating ideas from Chajewska's work are a definite

possibility for future work.

7.8 Murray and van Lehn

Murray and van Lehn's DT-Tutor (Murray and Lehn, 2000) uses a decision-theoretic method for designing an intelligent tutoring system. Actions within the tutorial dialogue are selected by the tutor, by choosing the available action with the highest expected utility. Utilities are determined by considering such factors as the "student's problem-related knowledge, focus of attention, independence, and morale, as well as action relevance and dialog coherence."

The tutoring system is implemented using a dynamic decision network. They introduce the idea of tutor action cycle networks (TACNs), which consist of "deciding a tutorial action and carrying it out, observing the next student action, and updating the student model based on these two actions" (Murray and Lehn, 2000).

This work is quite closely related to the work in this thesis. The main differences are that they have committed to the dynamic decision network framework and that they focus on one particular application area (intelligent tutoring). In our work, we have described an alternative to standard decision-theoretic approaches in situations where they will not work due to the unavailability of utility functions, and we have also developed a framework that is application-independent.

7.9 Bohnenberger and Jameson

Bohnenberger and Jameson have done significant work on modeling system-user interactions as Markov decision processes. In their recent article (Bohnenberger et al., 2002), they look at the problem of presenting personalized navigation advice to users as they carry out a shopping task in a simulated shopping mall. In an earlier paper (Jameson et al., 2001), they consider the problem of presenting instructions to a user on how to select options from a printing dialog box. This system attempts to model time pressure and the user's cognitive load, in order to make decisions about the trade-off between task performance and efficiency. They also discuss the fact that their problems are more accurately modeled as partially observable MDPs (Bohnenberger, 2002) and that much of their future work will focus on trying to deal with the computational intractability of the POMDP approach.

This work is quite closely related to the research described in this thesis. We also model the importance of time pressure, in an effort to balance task performance and efficiency. Their consideration of the user's cognitive load is also related to our bother factor, but we focus on the user's *general* attitudes toward interaction, as well as the system's perception of the user's knowledge, rather than focusing on the user's current activities. Again, the Bohnenberger and Jameson work is committed to the MDP framework, whereas we consider situations in which this methodology might not be the most appropriate.

7.10 Anderson et al.

For applications such as scheduling and routing tasks, Anderson et al. (2000) have investigated the concept of human-guided simple search. As with mixed-initiative systems, the idea behind this work is that the computer and a human user are working together on a task. However, they do not characterize their system as exhibiting mixed initiative, since the "user is always in control, and the computer has no representation of the user's intentions and abilities."

In human-guided simple search, the idea is that the roles of the system and user are clearly divided. The system is responsible for navigating through a search space, looking for local minima using a standard hill-climbing algorithm. The user, meanwhile, is responsible for identifying regions of the search space that look to be the most promising and for helping the system to avoid local, but non-global, minima.

For example, Anderson et al. (2000) describe a vehicle routing problem in which the goal is to minimize the number of trucks needed to deliver items to customers in different locations, within a certain time limit. If the system provides the user with a visual representation of the problem, then the user might be able to suggest moves such as assigning certain customers to certain trucks.

This work is relevant primarily because the authors have stated that their system does *not* qualify as a mixed-initiative system, but that they would be interested in investigating this possibility in the future. The models presented in this thesis would be a strong starting point for developing decision procedures for when to bring the user on board in such a system, beyond the clearly-defined role that is

currently specified for users.

7.11 Related user modeling work

In addressing the problem of specifying when a system should take the initiative, in a mixed-initiative artificial intelligence system, we are advocating a particular strategy for employing user models. Our algorithm clearly relies on having a user model which indicates information about a user's knowledge and attitudes. Other research in user modeling has studied how to employ user modeling information such as this for the purpose of tailoring output to users. Moreover, this related work has gone beyond simply suggesting a change in vocabulary, depending on the user's background. Paris (1991), for instance, examines how an entire scenario of explanation may differ, depending on whether the user is novice or expert. Carberry et al. (1999) highlight the difference between preferences and goals in the variation of output to a user. Ardissono and Sestero (1996) study when to issue clarifying questions for a particular user, based on an evaluation of the user and the current task. Raskutti and Zukerman (1994) examine what to say in light of a nuisance factor for the user. We are advocating a novel use for user models, namely to determine whether or not to interact with a user, depending on the perception of the user as indicated in the user model. This is distinctly different from altering the form or content of the generation based on information about the user.

Chapter 8

Conclusions

8.1 Thesis Summary

This thesis has presented computational models for the design of mixed-initiative artificial intelligence systems that are able to make rational decisions about interaction with potentially helpful users. One of the key challenges in designing mixed-initiative systems is providing the system with a decision procedure for determining *if* and *when* it should take the opportunity to solicit additional assistance from a potentially helpful user. The thesis provides designers of mixed-initiative systems with a systematic approach for constructing systems that can reason in a principled way about interaction with the user, regardless of the area of application of the system.

The approach was to integrate modeling of the user, the task and the dialogue simultaneously. Decisions about interaction were based on whether the perceived benefits of communication exceed the expected costs. Benefits are computed by

estimating by how much the system's performance on a task would improve if it were to obtain further information from a user. Cost factors include the time that will be required for the interaction and the degree to which the user is expected to be bothered by the system's interruption.

In particular, this thesis has emphasized the value of user modeling in mixed-initiative systems. Decisions about interaction should be based on a careful evaluation of the needs, preferences and abilities of the individual user, leading to mixed-initiative systems that are user-specific and therefore result in greater overall user satisfaction.

8.1.1 Summary of Contributions

The main contributions of this thesis are listed below.

- A systematic decision procedure was described for designers of mixed-initiative systems to use in determining what model would be most appropriate for a given application domain. Detailed models were presented for simple decision-theoretic agents, agents using Markov decision processes and agents using information-theoretic measures when certain information is unavailable. Conditions were identified under which each of these models would be suitable. In particular, we discussed the limitations of the Markov decision process model as a technique for modeling human-computer dialogues.

- Various factors were identified that must be modeled in *any* application domain in order for mixed-initiative systems to make rational decisions about interacting with users. Of particular importance were user modeling factors

such as the probability that a user will be able to answer a particular question and the willingness of a user to participate actively in a problem-solving process. Both of these factors have been neglected in earlier decision-theoretic research.

- An important contribution of this work was the careful study of the bother factor and its role in decisions about interaction with users. Different users have different preferences for the degree to which they wish to be involved in collaborative problem solving with a system. By considering the user's willingness to interact in estimating the expected bother that will result from an interaction, we allow for the development of systems that can better tailor their decisions about interaction to each individual user.

- In Section 3.8, a matrix-based model was developed to aid systems in determining the likelihood of users knowing the answer to a particular question. The approach was based on the idea of modeling not only the system's beliefs about a specific user's knowledge on a specific topic, but about that user's knowledge in more general topic areas and about the domain in general, as well as the perceived knowledge of users in the same class as the current user and of *all* users of the system. The method incorporates evidence from all of those different categories of knowledge and relies on the maintenance of several matrices containing weights. This is an important contribution to the user modeling community, as such a method has not been fully developed in the past.

- This thesis has also helped to emphasize the need for user modeling in the design of mixed-initiative systems. An effective mixed-initiative system must be flexible enough to keep control in certain situations and to yield control to the user at other times. It is essential that such a system be user-specific: the distribution of responsibilities that works best with one user might be inappropriate for another user. User modeling techniques must be incorporated into any system that can adapt its behaviour to each individual user.

8.2 Future Work

Several possibilities for future research were raised in Section 6.1. These and a few others are summarized below.

- **Understandability:** In this thesis, we emphasized the importance of modeling the probability that an interaction with a user would in fact be successful. One important reason that an interaction might not be successful is the possibility that the user might not be able to understand the question. This might be because of the terminology being used or simply because the user has not been involved enough in the problem-solving process and does not have the necessary contextual information to understand what is being asked. This topic was discussed only briefly in Section 6.1.1 and must be studied more extensively before being incorporated into the model.

- **User Availability and User Attention:** Also of possible interest to some designers are the ideas of user availability and user attention. If a system has

reliable information that a user is not currently available, or has reason to believe that the user is very unlikely to be present, then the expected benefit of interacting with this user is significantly decreased. Similarly, if the user is known to be distracted or involved in important unrelated work, the cost of interrupting the user should increase. These topics will also be investigated in more depth in future work.

- **Predictability:** Another interesting topic for future work is the idea of predictability mentioned in Section 6.1.2. The idea again is that, all other things being approximately equal, a system should have a tendency to do what the user might expect it to do, based on previous interactions. Research must be done to determine if users really do prefer to have systems behave in a consistent manner, even if it might mean that the system is not behaving rationally in some cases.

- **Further developing the heuristic functions:** As discussed in Section 6.1.6, it will often be necessary for systems to use heuristics when it is not possible to estimate the utilities of possible future states or even to enumerate the possible answers that might be provided by users when asked questions by the system. These heuristics are based on the amount of experience that the system has in working with individual users and in the problem-solving domain in general, as well as on any information that the system might have received about its previous performance on the task. The preliminary heuristics provided in Section 6.1.6 must be further developed before they can be implemented in a full-fledged mixed-initiative system.

- **Multiple users:** A very interesting future extension of this work is to generalize the model so that it will deal with systems that have the option of interacting with many different users. The goal of deciding which user to choose would be accomplished by considering the benefits and costs involved in interacting with each available user. This extension makes the overall research problem richer and more challenging.

- **Practical applications:** Another very important area for future work is to develop more significant implementations of the model and to test the resulting systems on practical applications with real users. Of particular importance here is the idea of getting a more reliable measure of the cost of bothering the user. The bother models presented in this thesis are useful guides for programmers wishing to design systems that take bother cost into account when deciding about interaction. However, performing surveys with actual users to elicit their true opinions about the trade-off between task success and the costs of interruption will be beneficial.

- **Extended experimentation:**

  Additional experimentation with the existing models would also be very useful. For the short term, it would be useful to conduct further experiments in which different users have different actual probabilities of providing wrong answers, beyond the cases presented in Chapter 5. Also, experiments in which agents modify their P_{UK} value as the interaction progresses, rather than assuming the same P_{UK} throughout the session, would be a valuable variation to explore.

For the long term, experiments should be modified to deal with one of the real-world domains discussed in this thesis, such as scheduling, translation or book purchasing, compared to the tasks presented in Chapter 5, which were generic. This would involve implementing several agents using different values for the various parameters in the model and testing their performance, in terms of overall task performance and user satisfaction.

Also, agents working on real-world tasks will have more than three actions to consider at any one time and more than three possible questions to ask the user, as was the case with the experiments in Chapter 5. It will be important to investigate how well this model will scale up to larger problems.

Furthermore, the utility functions presented in Chapter 5 were chosen randomly. In real-world situations, actual users will have utility functions with more structure.

Other useful future experimentation would involve comparing our systems to existing mixed-initiative systems that have been designed without our models. Appropriate criteria for evaluation would include task success and user satisfaction.

- **The question-generating module:** As acknowledged in Section 6.1.5, the current models do not provide very complex modeling of situations in which follow-up questions might be necessary after asking users certain questions. This will be an important consideration in future versions of these models. In addition, future work may involve admitting possible faults in the questions being generated and adjusting calculations accordingly. As a starting point,

we would explore detailed examples of such cases.

- **Intelligent tutoring applications:** One particular domain of interest for the potential application of our models is that of intelligent tutoring systems. As mentioned in Section 2.1.3, there are several properties that make this application domain a unique one. Most notably, in intelligent tutoring systems, the overall evaluation of a course of action taken by the system must include an evaluation of how well the user has actually *learned* from the experience. This can be difficult to judge and makes the overall problem of assigning utilities to outcomes much more challenging. For future work, it would be useful to provide additional insights into how to develop these utility functions, in order to employ our model in this application area.

- **Adjustable autonomy research:** Further work must be devoted to examining the connection between the work in this thesis and research in the adjustable autonomy community. Adjustable autonomy research involves systems that can modify the degree to which they will act autonomously on a particular task. Essentially, the idea of giving up autonomy is equivalent to the idea of asking for additional help, whether it be from a human user or from another artificial agent. In the case of adjustable autonomy systems, the help might not come in the form of additional information, but rather in the form of guidance as to what steps to perform next in a problem-solving process. There is a definite need for researchers in adjustable autonomy and mixed-initiative interaction to work together in the future, as the two problems are quite closely related, and the role of communication among the collaborating

parties is crucial in both areas.

8.3 Final comments

Mixed-initiative interaction is gaining recognition in artificial intelligence research as an important component of systems designed for use in application areas as diverse as robotics, military planning, intelligent tutoring and trip scheduling. An important challenge in designing these systems is to specify when the system should take the initiative to interact with the user, in order to optimize both the performance of the system and the satisfaction of the user. By providing system designers with a systematic approach for constructing systems that can reason in a principled way about interaction with the user, regardless of the area of application of the system, this thesis has made a substantial contribution to mixed-initiative research.

References

Aist, G. (1997). Challenges for a mixed initiative spoken dialog system for oral reading tutoring. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 1–6. AAAI Press.

Allen, J. (1994). Mixed-initiative planning: Position paper. Presented at the ARPA/Rome Labs Planning Initiative Workshop, 1994.

Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K. (2000). Human-guided simple search. In *Proceedings of AAAI-2000*, pages 209–216. AAAI Press.

Ardissono, L. and Goy, A. (2000). Tailoring the interaction with users in web stores. *User Modeling and User-Adapted Interaction*, 10(4):251–303.

Ardissono, L. and Sestero, D. (1996). Using dynamic user models in the recognition of the plans of the user. *User Modeling and User Adapated Interaction*, 5(2):157–190.

Bacchus, F., Grove, A., Halpern, J., and Koller, D. (1994). Forming beliefs about a changing world. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 222–229, Menlo Park, California. AAAI Press.

Bauer, M., Dengler, D., Meyer, M., and Paul, G. (2000). Instructible information agents for web mining. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces.*

Bellman, R. E. (1957). *Dynamic Programming.* Princeton University Press.

Billsus, D. and Pazzani, M. (1999). A hybrid user model for news story classification. In *Proceedings of UM'99, Banff, Alberta, Canada*, pages 99–108.

Bohnenberger, T. (2002). Decision-theoretic planning for intelligent user interfaces. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Doctoral Consortium.*

Bohnenberger, T., Jameson, A., Krueger, A., and Butz, A. (2002). Location-aware shopping assistance: Evaluation of a decision-theoretic approach. In *Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices (Mobile-HCI-02).*

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *Proceedings of IJCAI-99*, pages 478–485.

Brainov, S. and Hexmoor, H. (2003). *Agent Autonomy (Hexmoor, Falcone and Castelfranchi, eds.)*, chapter 4. Quantifying Autonomy. Kluwer Publishers.

Brusilovsky, P. and Maybury, M. T. (2002). From adaptive hypermedia to the adaptive web. *Communications of the ACM*, 45(5):30–33.

Buczak, A., Zimmerman, J., and Kurapati, K. (2002). Personalization: Improving ease-of-use, trust and accuracy of a tv show recommender. In Ardissono, L. and Buczak, A., editors, *Proceedings of the AH 2002 Workshop on Personalization in Future TV, Málaga, Spain*, pages 9–18.

Burstein, M. and McDermott, D. (1996). Issues in the development of human-computer mixed-initiative planning. In Gorayska, B. and Mey, J., editors, *In Search of a Humane Interface*, pages 285–303. Elsevier Science B.V.

Carberry, S. (1989). Plan recognition and its use in understanding dialog. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*, pages 133–162. Springer-Verlag.

Carberry, S. (1997). Discourse initiative: Its role in intelligent tutoring systems. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction, Stanford, CA*, pages 10–15. AAAI Press.

Carberry, S., Chu-Carroll, J., and Elzer, S. (1999). Constructing and utilizing a model of user preferences in collaborative consultation dialogues. *Computational Intelligence*, 15(3):185–217.

Carletta, J. C. (1996). Assessing the reliability of subjective codings. *Computational Linguistics*, 22(2):249–254.

Chajewska, U., Koller, D., and Parr, R. (2000). Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369.

Chi, E. H., Pirolli, P., Chen, K., and Pitkow, J. E. (2001). Using information scent to model user information needs and actions and the web. In *Proceedings of CHI 2001*, pages 490–497.

Chin, D. and Crosby, M. (2002). Introduction to the special issue on empirical evaluation of user models and user modeling systems. *User Modeling and User-Adapted Systems*, 12(2–3):105–109.

Chin, D. N. (1989). KNOME: Modeling what the user knows in UC. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*, pages 74–107. Springer-Verlag.

Chu-Carroll, J. and Brown, M. (1997). Initiative in collaborative interactions – its cues and effects. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 16–22. AAAI Press.

Cohen, R., Allaby, C., Cumbaa, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., Qian, A., and Siddiqi, S. (1998). What is initiative? *User Modeling and User-Adapted Interaction*, 8(3–4):171–214.

Cohen, R. and Fleming, M. (2003). *Agent Autonomy (Hexmoor, Falcone and Castelfranchi, eds.)*, chapter 7. Adjusting the autonomy in mixed-initiative systems by reasoning about interaction. Kluwer Publishers.

Cohen, R., Schmidt, K., and van Beek, P. (1994). A framework for soliciting clarification from users during plan recognition. In *Proceedings of the Fourth International Conference on User Modeling*, pages 11–17.

Cook, R. and Kay, J. (1994). The justified user model: a viewable, explained user model. In *Proceedings of the 4th International Conference on User Modeling*,

pages 73–78.

Cox, M., editor (1999). *Proceedings of the 1999 AAAI Workshop on Mixed-Initiative Intelligence.* Menlo Park, CA: AAAI Press.

Cox, M. and Veloso, M. (1997). Controlling for unexpected goals when planning in a mixed-initiative setting. In *Proceedings of the 8th Portuguese AI Conference, Coimbra, Portugal*, pages 309–318.

Feinberg, E. and Shwartz, A., editors (2001). *Handbook of Markov Decision Processes: Methods and Applications.* Kluwer.

Ferguson, G. and Allen, J. (1998). TRIPS: An intelligent integrated problem-solving assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI*, pages 567–573.

Ferguson, G., Allen, J., and Miller, B. (1996). TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligent Planning Systems (AIPS-96), Edinburgh, Scotland*, pages 70–77.

Fleming, M. (1998). Designing more interactive interface agents. Master of Mathematics thesis, University of Waterloo, Waterloo, Ontario.

Fleming, M. and Cohen, R. (2000). System initiative influenced by underlying representations in mixed-initiative planning systems. In *AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems*, pages 18–21.

Freedman, R. (1997). Degrees of mixed-initiative interaction in an intelligent tutoring system. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 44–49. AAAI Press.

Gmytrasiewicz, P. J. and Durfee, E. H. (1995). A rigorous, operational formalization of recursive modeling. In *Proceedings of the First International Conference on Multiagent Systems, ICMAS '95*, pages 125–132.

Gmytrasiewicz, P. J. and Durfee, E. H. (2001). Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 4(3):233–272.

Godden, K. (2000). The evolution of CASL controlled authoring at general motors. In *Proceedings: The 3rd International Workshop on Controlled Language Applications*.

Grayson, C. (1960). Decisions under uncertainty: Drilling decisions by oil and gas operations. Technical report, Division of Research, Harvard Business School, Boston.

Grosz, B. and Sidner, C. (1986). Attention, intentions and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Grosz, B. J. and Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357.

Grosz, B. J. and Sidner, C. L. (1990). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA.

Guinn, C. (1996). Mechanisms for mixed-initiative human-computer collaborative discourse. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 278–285.

Haddawy, P. and Hanks, S., editors (1998). *Papers from the 1998 AAAI Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems.* AAAI Press.

Haller, S., Kobsa, A., and McRoy, S., editors (1999). *Computational Models of Mixed-Initiative Interaction.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Haller, S. and McRoy, S., editors (1997). *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction.* AAAI Press.

Hexmoor, H., Falcone, R., and Castelfranchi, C., editors (2003). *Agent Autonomy.* Kluwer Publishers.

Horvitz, E. (1988). Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning, Special Issue on Uncertainty in Artificial Intelligence*, 2:247–302.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems,*

Pittsburgh, PA, pages 159–166. ACM Press.

Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-sensitive alerting. In *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence*, pages 305–313.

Horvitz, E. and Paek, T. (2001). Harnessing models of users' goals to mediate clarification dialog in spoken language systems. In *Proceedings of the Eighth International Conference on User Modeling*, pages 3–13.

Horvitz, E. and Rutledge, G. (1991). Time-dependent utility and action under uncertainty. In *Uncertainty in Artificial Intelligence*, pages 151–158. Morgan Kaufman.

Howard, R. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, SSC-2:22–26.

Jackson, W. and Havens, W. (1995). Committing to user choices in mixed initiative csps. In *Proceedings of the Fifth Scandinavian Conference on Artificial Intelligence, Trondheim, Norway.* Also appears as Simon Fraser University School of Computing Science Technical Report CMPT TR 95-1.

Jameson, A., Großmann-Hutter, B., March, L., Rummer, R., Bohnenberger, T., and Wittig, F. (2001). When actions have consequences: Empirically based decision making for intelligent user interfaces. *Knowledge-Based Systems*, 14:75–92.

Kamm, C., Litman, D., and Walker, M. A. (1998). From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP98)*, pages 1211–1214.

Kass, R. (1989). *User Models in Dialog Systems (Kobsa and Wahlster, eds.)*, chapter 14: "Student Modeling in Intelligent Tutoring Systems – Implications for User Modeling", pages 386–410. Springer-Verlag.

Kass, R. (1991). Building a user model implicitly from a cooperative advisory dialog. *User Modeling and User-Adapted Interaction*, 1(3):203–258.

Kass, R. and Finin, T. (1988). Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22.

Kautz, H., editor (1998). *Papers from the AAAI-98 Workshop on Recommender Systems.* AAAI Press.

Kautz, H. and Allen, J. (1986). Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37.

Keeney, R. and Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* Wiley, New York.

Keim, G., Fulkerson, M., and Biermann, A. (1997). Initiative in tutorial dialogue systems. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction, Stanford, CA*, pages 85–88. AAAI Press.

Kobsa, A. and Wahlster, W., editors (1989). *User Models in Dialog Systems.* Springer-Verlag.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Kortenkamp, D., Bonasso, R., Ryan, D., and Schreckenghost, D. (1997). Traded control with autonomous robots as mixed-initiative interaction. In *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 89–94. AAAI Press.

Kozierok, R. (1993). A learning approach to knowledge acquisition for intelligent interface agents. Master of Science thesis, Massachusetts Institute of Technology, Cambridge MA.

Lambert, L. and Carberry, S. (1991). A tripartite plan-based model of dialogue. In *Proceedings of the 29th ACL*, pages 47–54.

Lester, J., Callaway, B., Stone, B., and Towns, S. (1997). Mixed initiative problem solving with animated pedagogical agents. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 98–104. AAAI Press.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using Markov decision process for learning dialogue strategies. In *Proceedings of ICASSP 98.*

Lieberman, H. (1995). Letizia: An agent that assists Web browsing. In *Proceedings of IJCAI '95*. AAAI Press.

Litman, D. and Pan, S. (1999). Empirically evaluating an adaptable spoken dialogue system. In Kay, J., editor, *User Modeling – Proceedings of the Seventh International Conference, Banff, Alberta*, pages 55–64.

Litman, D., Pan, S., and Walker, M. (1998). Evaluating response strategies in a web-based spoken dialogue agent. In *Proceedings of ACL/COLING 98*, pages 780–786.

Litman, D. J., Kearns, M. S., Singh, S., and Walker, M. A. (2000). Automatic optimization of dialogue management. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*.

Litman, D. J. and Pan, S. (2000). Predicting and adapting to poor speech recognition in a spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 722–728.

Lochbaum, E. and Sidner, C. (1990). Models of plans to support communication: An initial report. In *Proceedings of AAAI-90*.

McRoy, S. W. and Ali, S. S. (1999). A practical, declarative theory of dialog. In Cox, M., editor, *Proceedings of the 1999 AAAI Workshop on Mixed-Initiative Intelligence*. Menlo Park, CA: AAAI Press.

Murray, R. and Lehn, K. V. (2000). DT Tutor: A decision-theoretic, dynamic

approach for optimal selection of tutorial actions. In *Intelligent Tutoring Systems, 5th International Conference*, pages 153–162.

Myers, K. (1998). Towards a framework for continuous planning and execution. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning.* AAAI Press.

Myers, K. (2000). Domain metatheories: Enabling user-centric planning. In *Proceedings of the AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems.*

Myers, K., Jarvis, P., Tyson, W., and Wolverton, M. (2003). A mixed-initiative framework for robust plan sketching. In *Proceedings of the 2003 International Conference on Automated Planning and Scheduling.*

Myers, K. and Morley, D. (2001). Directing agent communities: An initial framework. In *IJCAI-01 Workshop on Autonomy, Delegation and Control.*

Myers, K., Tyson, W., Wolverton, M., Jarvis, P., Lee, T., and desJardins, M. (2001). Passat: A user-centric planning framework. In *Third International NASA Workshop on Planning and Scheduling for Space.*

Novick, D. and Sutton, S. (1997). What is mixed-initiative interaction? In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 114–116. AAAI Press.

Paek, T. and Horvitz, E. (1999a). Uncertainty, utility, and misunderstanding: A decision-theoretic perspective on grounding in conversational systems. In

AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems.

Paek, T. and Horvitz, E. (1999b). Uncertainty, utility, and misunderstanding: A decision-theoretic perspective on grounding in conversational systems. In *AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems.*

Paris, C. (1991). The role of the user's domain knowledge in generation. *Computational Intelligence*, 7(2):71–93.

Pazzani, M. J., Muramatsu, J., and Billsus, D. (1996). Syskill and webert: Identifying interesting web sites. In *AAAI/IAAI, Vol. 1*, pages 54–61.

Pollack, M. (1990). *Intentions in Communication*, chapter Plans as complex mental attitudes. MIT Press.

Puterman, M. L. (1994). *Markov Decison Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc.

Raskutti, B. and Zukerman, I. (1994). Query and response generation during information-seeking interactions. In *Proceedings of UM'94, Hyannis, Mass*, pages 25–30.

Raskutti, B. and Zukerman, I. (1997). Generating queries and replies during information-seeking interactions. *International Journal of Human Computer Studies*, 47(6):689–734.

Rich, C. and Sidner, C. (1997a). Collagen: When agents collaborate with people. In *First International Conference on Autonomous Agents*, pages 284–291.

Rich, C. and Sidner, C. (1997b). Segmented interaction history in a collaborative agent. In *Third International Conference on Intelligent User Interfaces*, pages 23–30.

Rich, C. and Sidner, C. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3–4):315–350.

Rich, E. (1989). Stereotypes and user modelling. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*. Springer-Verlag.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc.

Schuurmans, D. and Patrascu, R. (2001). Direct value-approximation for factored mdps. In *Advances in Neural Information Processing Systems 14*.

Shah, F. and Evens, M. (1997). Student initiatives and tutor responses in a medical tutoring system. In Haller, S. and McRoy, S., editors, *Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction*, pages 138–144. AAAI Press.

Shannon, C. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.

Shifroni, E. and Shanon, B. (1992). Interactive user modeling: An integrative explicit-implicit approach. *User Modeling and User-Adapted Interaction*,

2(4):331–336.

Siegel, S. and Castellan, N. (1988). *Nonparametric Statistics for the Behavioral Sciences*. McGraw Hill.

Smith, R. and Hipp, D. (1994). *Spoken Natural Language Dialog Systems – A Practical Approach*. Oxford University Press.

Sullivan, D., Grosz, B., and Kraus, S. (2000). Intention reconciliation by collaborative agents. In *Proceedings of ICMAS-2000*, pages 293–300. IEEE Computer Society Press.

Tecuci, G., Boicu, M., Wright, K., and Lee, S. (1999). Mixed-initiative development of knowledge bases. In *Papers from the AAAI-99 Workshop on Mixed-Initiative Intelligence, Orlando, Florida*, pages 51–59.

van Beek, P., Cohen, R., and Schmidt, K. (1993). From plan critiquing to clarification dialogue for cooperative response generation. *Computational Intelligence*, 9(2):132–154.

Veloso, M., Mulvehill, A., and Cox, M. (1997). Rationale-supported mixed-initiative case-based planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), Providence, RI*, pages 1071–1077.

Walker, M., Hindle, D., Fromer, J., Fabbrizio, G. D., and Mestel, C. (1997a). Evaluating competing agent strategies for a voice email agent. In *Proceedings of the 5th European Conference on Speech Technology and Communication (EUROSPEECH 97)*, pages 2219–2222.

Walker, M., Litman, D., Kamm, C., and Abella, A. (1997b). PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, pages 271–280.

Walker, M. and Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th Annual Meeting of the ACL (ACL-90), Pittsburgh, PA*, pages 70–76.

White, D. (1993). *Markov Decision Processes*. John Wiley & Sons.

Wu, D. (1991). Active acquisition of user models: Implications for decision-theoretic dialog planning and plan recognition. *User Modeling and User-Adapted Interaction*, 1(2):149–172.

Xuan, P., Lesser, V., and Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: model and experiments. In Müller, J. P., Andre, E., Sen, S., and Frasson, C., editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 616–623, Montreal, Canada. ACM Press.

Zhou, R. and Hansen, E. (2001). An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 707–714.