A Computational Framework for Mixed-Initiative Dialog Modeling

陳淑芳
CHAN, Shuk Fong

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

August 2002

Abstract

This thesis aims to use different computational and representational frameworks to model mixed-initiative interactions. We first incorporate our dialog model in ISIS, a trilingual spoken dialog system for the stocks domain, with a form-based approach. This mixed-initiative dialog model is capable of inheriting selected discourse history, combining online interaction with offline delegation sub-dialogs, as well as learning new information through a conversation to expand the system's knowledge space. We used declarative templates for dialog turn management in ISIS. However, we found that the type of mixed-initiative dialogs in ISIS may be too simplistic when compared to real interactions. Hence, we study the interdependencies among dialog acts, task goals and discourse inheritance in human-human mixed-initiative dialogs in the CU Restaurants domain. We have developed a selective category inheritance strategy, which outperformed two control strategies where none or all of the categories are inherited. We then leverage from our analysis results in the CU Restaurants domain to a rule-based dialog system. Additionally, we have applied Belief Networks to automate identification of task goals and dialog acts based on the input categories. Evaluation indicates that our system achieved promising results on both user satisfaction and task completion.

摘要

本論文試圖以不同的電腦流程及表示方法去模塑混合對答 (mixed-initiative) 的對話模式。 我們首先將混合對答對話模式應用於ISIS－一個以三語爲本及以股票實時資訊及買賣 (real-time stock market information and transactions) 爲特定領域的人機對話系統。 當中，我們在ISIS內容上建立了兩個新研究方向，分別是自動化合併未知的股票名稱於 系統的知識庫及在單一的對話線上在線對話 (online interaction) 及之前已授權的 服務完成通知 (offline delegation) 的互相切換。爲了令建立的人機對話中更能配合 用户的需要，我們利用在飲食業服務 (restaurants) 領域中所收集的人人 (human-human) 對話研究及擷取顧客在索取服務時的目的 (task goal)，意向 (dialog act) 及語意內容延續的 因果關係。當中，我們採用了信念網絡 (Belief Networks) 去推斷顧客在索取服務時的目的及意向。 基於我們研究的結果，我們建立了一個以飲食業服務爲特定領域的人機對話應用系統。在基準測試當中 顯示我們的系統在用户的滿意度上有合理的表現。

Acknowledgments

First, I would like to sincerely thank my supervisor, Professor Helen Meng for her guidance and support in my research. In the past four years, I am grateful for being her fyp student, Project Engineer as well as her master student. During the darkness, she supported me to carry on. Now, I still remember how many versions Helen helped me to modify my HLT paper. While I have some improvement in my research, she shared the joy with me. Even with the busy schedule as the group leader, she still always squeezed an extra minute to share and talk to me as a friend. It has been a great pleasure and honor to learn from her and work closely with her.

Next, I wish to thank the members of my thesis committee, Professor Stephanie Seneff from MIT; Professor Wai Lam and Professor Jeffrey Yu at the Department of SEEM, for their precious comments and interest.

As a member of the Human-Computer Communications Laboratory, I would like to thank all my office mates. A special thanks to Kin, who read this thesis and gave me excellent suggestions and useful feedback on my research. Thanks to Tiffany and Yuk for their friendships and support. Thanks to Fan for providing technical assistance on the Belief Networks. Thanks to Tony, Ada Luk, Julia, Chat, Michael Lo, Homa, Sally, Connie, Carmen and Timmy for their caring and sharing. Thanks to Mandy, Sunny, Silvia, Ida, Angie,

Cindy, Cecia, Bonnie, May and Ma Bin for the mutual encouragement during the hard time of the study. I also want to thank the ISIS team: Julia and Chat, for their help in the project.

I would like to thank my dearest God. Thanks to Him for accompanying with me and helping me all the time.

Finally, I would like to express my very deep gratitude to Groundmouse, my Mom and Dad for all of their love, understanding, humor, caring and support. My Mom and Dad do not know what my research has been, but they love me the same forever. Also, I would like to say a special thanks to Groundmouse for his enduring support and care. I love them very much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Computers are ubiquitous in our lives. Nowadays, people can do everything with computers via the Internet, such as book-ordering, on-line shopping, billing, etc. Among the various means of communication, speech is the most *natural* and *common* among humans. Thus, it should be desirable to develop computer systems that can communicate with humans via speech. A Spoken Dialog System (SDS) is a computer system that supports conversations between humans. SDSs integrate a plethora of spoken language technologies, including speech recognition, language understanding, dialog management, speech synthesis, etc. In a SDS, speech dialog can be depicted as a closed loop system with separate layers of interactions, where independent modules perform the transformation between these layers. Figure 1.1 [1] shows the separation and the modules associated with each separate layer.

As depicted from Figure 1.1 [1], the speech dialog between a user and an agent can be classified into three different levels of interactions: speech

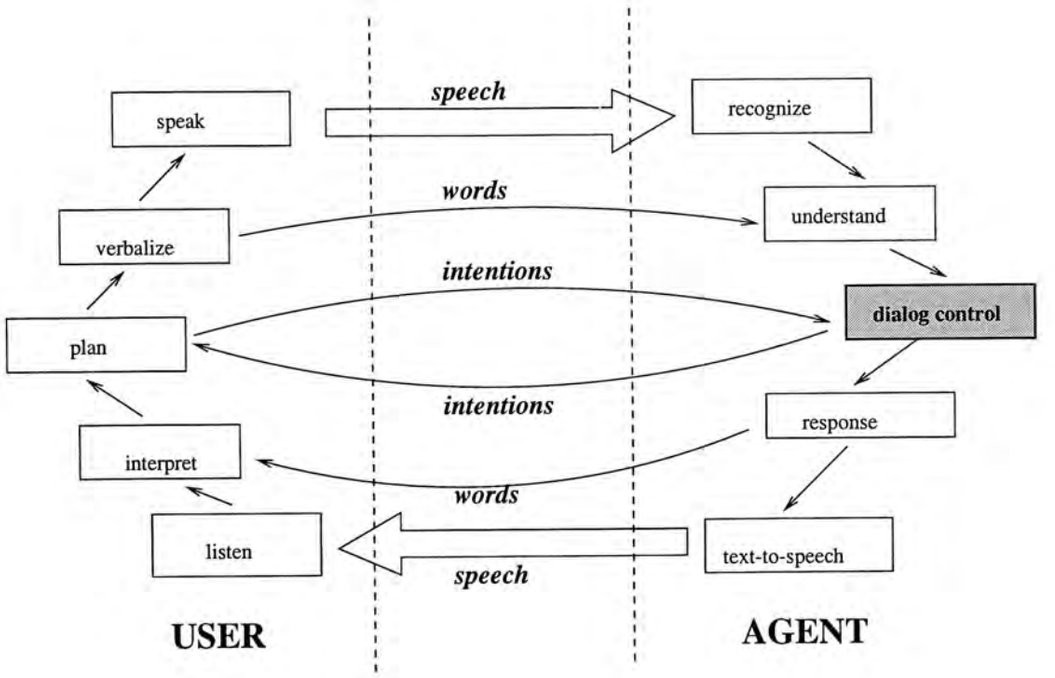

Figure 1.1: Conversational interaction between a user and an agent. This figure is referenced from [1].

signals, word sequences and intentions. Speech is carried by a set of acoustic signals. Word sequences represents the next level decoding the speech signals into a clean text. Intentions cannot be observed directly but can be viewed as the actual information. In each level of interaction, each module is associated with a transformation step. At the beginning of the interaction, both the user and the agent try to figure out the words from the received speech through the modules *listen* and *recognize*. Then, they *interpret* and *understand* the words. After they get the intentions from the words, they consider how they should respond through the modules *plan* and *dialog control*. In this level, there is a transformation from the speaker's intentions to the listener's intentions. Then, the user and the agent convert the planned intentions into words through the modules *verbalize* and *response*. Finally, based on the

modules *speak* and *text-to-speech*, they answer via speech.

Dialog management, which manages the *dialog control*, is one of the key processes in a SDS. It is the source of the system's intelligent behavior. Dialog management concerns how the system interacts with the user. The dialog model in a SDS is the most critical component for the system's usability. It determines what the users are able to request to the system, in which way and at what time during the dialog session. There are three common types of dialog models, including system-initiative dialog model, user-initiative dialog model and mixed-initiative dialog model.

System :	*"Tell me the stock name for which you want to check the news."*
User :	*"Cheung Kong."*
System :	*"Tell me the dates of the news you want."*
User :	*"Yesterday."*
System :	*"Here is the news of Cheung Kong Holdings Limited from yesterday."*

Table 1.1: An example of a system-initiative dialog.

In a *system-initiative* dialog model, only the *system* can initiate and control the dialog flow over the course of conversation. Table 1.1 shows an example session of system-initiative interactions. Note that in the first two dialog turns, the system presents directed questions and guides the user to provide the information of stock name and the date. The cooperative user follows the system and replies *"Cheung Kong"* and *"Yesterday"* accordingly. The system acts actively while the user answers the questions passively. The system-initiative dialog model is usually implemented as a tree structure. Since system-initiative interactions always acquire one attribute in each dialog turn; it can attain a high task completion rate. However, the user may

feel frustrated as a result of the inflexible dialog flow.

User	: *"I would like to check yesterday's news about Cheung Kong."*
System	: *"Here is the news of Cheung Kong Holdings Limited from yesterday."*

Table 1.2: An example of a user-initiative dialog.

In a *user-initiative* dialog model, only the *user* is in control of the flow of conversation. As observed in Table 1.2, the user initiates the dialog by saying *"I would like to check yesterday's news about Cheung Kong."*. The system attempts to retrieve relevant information and present the result to the user. User-initiative dialog modeling allows unconstrained input from the users. Since the system acts passively in user-initiative interactions, the user is often unaware of the limitations of the system. The user may feel frustrated if the system cannot respond to his expectations. As a result, in real applications, this model usually has lower task completion rate than the system-initiative dialog model.

User	: *"I would like to check yesterday's news."*
System	: *"Certainly, sir. What is the stock name you are looking for?"*
User	: *"Cheung Kong please."*
System	: *"Here is the news of Cheung Kong Holdings Limited from yesterday."*

Table 1.3: An example of a mixed-initiative dialog.

The *mixed-initiative* dialog model [11, 12] refers to a flexible interaction strategy, where both the human and computer can influence the conversational flow. Mixed-initiative interactions let both the human and computer work most effectively as a team. In order to achieve this, the computer should

be able to collaborate with the human to complete the task. Table 1.3 illustrates an example dialog for mixed-initiative dialog model. In the first dialog turn, the user asks for yesterday's news. The system detects that the stock name is missing for this request, so it asks the user for the information. Note that the system has the flexibility to relinquish initiative. The system can also decide when to ask the user for clarification.

Among these three dialog models, the *mixed-initiative dialog model* is deemed most desirable, since it provides greater flexibility and better potential in achieving both higher task completion rates as well as greater user satisfaction. In this thesis, we attempt to explore mixed-initiative interactions and use different representational and computational frameworks to model mixed-initiative dialogs.

1.2 Thesis Contributions

This thesis explores and investigates various representational and computational frameworks for mixed-initiative dialog modeling in specific domains, namely, stocks and restaurants. We begin with a system known as ISIS, which abbreviates *Intelligent Speech for Information Systems.* ISIS is a spoken dialog system which supports multi-modal and mixed-modal input. Input modalities include speaking, typing and mouse-clicking. Output media include synthesized speech, text, tables and graphics. The ISIS knowledge base is restricted to the stocks domain. ISIS resembles a virtual stockbroker that can provide the user with real-time quotes, personal portfolio information as well as help with transactions. Within the context of ISIS, we have designed and implemented a mixed-initiative dialog model capable of (i) in-

heriting selected discourse history, (ii) combining online interaction with offline delegation sub-dialogs, as well as (iii) learning new information through a conversation to expand the system's knowledge base. In particular, the notion of combined online/offline sub-dialogs (from point *ii*) can support both the speech and text user input while the notion of "learning sub-dialogs" (from point *iii*) can only support text input in the current approach, bypassing the issues that have to do with misrecognition of unknown words. Moreover, they are novel and are original contributions of this thesis. In ISIS, discourse inheritance is achieved by using an electronic form (E-form) model [13], while the response generation is controlled by the templates which state the mapping between the query cases and their corresponding response trigger words.

Example 1	
User:	*"I should buy HSBC."* **Task Goal (TG):** BUY **Dialog Act (DA):** REQUEST_ACTION **Semantics for TG:** {<Buy = 'buy'> <StockName = 'HSBC'>} **Semantics for DA:** {<ActionWord = 'buy'> } **Syntactics for DA:** {<Period = '.'>}
ISIS:	*"Please provide the bid price and the number of lots."* (Correct)
Example 2	
User:	*"Should I buy HSBC?"* **TG:** BUY **DA:** REQUEST_COMMENT **Semantics for TG:** {<Buy = 'buy'> <StockName = 'HSBC'>} **Semantics for DA:** {<RequestPhrase = 'Should I'> } **Syntactics for DA:** {<Quest = '?'>}
ISIS:	*"Please provide the bid price and the number of lots."* (Wrong)

Table 1.4: ISIS generated the same response to the users even though the queries are with different communicative intentions.

In mixed-initiative dialog modeling, the system should take the initiative whenever it can help the user to complete his task effectively. However, our experience in ISIS shows that the initiative shifts to the system only under several query conditions, including: prompting for missing information, invoking confirmation sub-dialogs and offering price alert services. Hence, the type of mixed-initiative dialogs in the ISIS domain may be too simplistic when compare to real interactions. In addition, the shifting of initiatives should depend on more than just the task goal. For example in Table 1.4, the user queries in example 1 and example 2 have the same semantics leading to the same *task goal.* However, these two queries contain similar words but have different communicative intentions stated in terms of *dialog acts.* Without considering the dependence of dialog acts, ISIS treated both queries as BUY requests and generated the same response to the users. In example 1, the ISIS response should be correct since the user is requesting a BUY action in the query (REQUEST_ACTION). However, in example 2, the response should be wrong since the user is actually requesting a comment (REQUEST_COMMENT), not an action. These examples indicate that both the *domain-dependent* task goals and *domain-independent* dialog acts should be captured in our dialog model. As a result, the second part of our investigation involves a study of the interdependencies among dialog acts, task goals and discourse inheritance in mixed-initiative dialogs in the restaurants domain. We analyzed a corpus of human-human mixed-initiative dialogs involving requests from the customer and responses from the waiter. In the framework of our study, communication for every request or response is characterized by its semantic and syntactic categories, dialog act(s) and task goal(s). As

the dialog progresses from one turn to the next, selected categories need to be inherited in the discourse and inheritance may be dependent on the task goal or dialog act. The inherited categories augment those in the current (context-dependent) request to help determine its task goal and dialog act. These results of our observations and analysis have been incorporated into a model for discourse inheritance, which is the key contribution of the second part of this thesis.

Aside from automatic analysis of the user's requests, an SDS also needs to respond in a mixed-initiative fashion. Hence the third part of our investigation relates to cooperative response generation in mixed-initiative dialog modeling. We leverage from our analysis results in the CU Restaurants domain to a rule-based dialog system, which generates cooperative response based on the identified task goals, dialog acts and categories. Additionally, we have applied Belief Networks to automate identification of task goals and dialog acts based on input categories. Table 1.5 illustrates an example dialog which is generated from our implementation framework, where the system gives recommendation cooperatively while the customer is ordering. This model for cooperative response generation in the CU Restaurants domain is the key contribution of the third part of this thesis.

User	: *"I would like to have a seafood platter, please."*
System	: *"Anything else, sir? Today our green salad is good."*
User	: *"No, thanks."*
System	: *"Thank you, sir. You have ordered a seafood platter. Your order should arrive within 15 minutes."*

Table 1.5: An example dialog generated from our implementation framework.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 describes previous work in dialog management of spoken dialog systems, studies of mixed-initiative dialogs and cooperative response generation. Chapter 3 details our representational and computational framework for mixed-initiative dialog modeling in ISIS. Chapter 4 presents our study of the structures in human-human mixed-initiative dialogs in the CU Restaurants domain. Chapter 5 demonstrates our computational model for generating the cooperative responses in the CU restaurants domain. Conclusions and future work are provided in Chapter 6.

Chapter 2

Background

This thesis explores mixed-initiative interactions. In spoken dialog systems, dialog management involves many challenges, especially in the development of representational and computational models for mixed-initiative interactions. In order to be effective in the development of dialog systems, we can understand more about the characteristics of human-human dialogs. While interacting with the users in the dialog systems, cooperative response generation should help in achieving better task completion as well as user satisfaction. In this chapter, we will study the background information in these areas. In Section 2.1, we will investigate the challenges and design concerns of mixed-initiative dialog models. In Section 2.2, we will describe different approaches for handling mixed-initiative interactions in various spoken dialog systems. In Section 2.3, we will illustrate the recent study of mixed-initiative structures in human-human dialogs and human-computer dialogs. Lastly in Section 2.4, we will present different approaches in cooperative response generation.

2.1 Mixed-Initiative Interactions

Research in mixed-initiative interactions has been going on for a decade. The definition [14] of the word "initiative" is to "control the flow of conversation". Hence, "mixed-initiative interactions" means interchanging the flow control between two agents throughout a conversation. In a human-computer conversational system with mixed-initiative dialog control, sometimes the human may take the control of the conversation and the computer responds to the user's requests. However, the roles may be reversed at other times. As referenced from [15], the challenge in handling mixed-initiative interactions is to define a computational model of how initiative should be controlled in a dialog. Burstein and McDermott [16] pointed out that the overall objective of the research of mixed-initiative interactions is to explore the strengths of both human and computer in order to build effective dialog plans more quickly and with greater reliability.

In developing a mixed-initiative dialog strategy, it is important to know when to control between the user and the system. Whittaker et al. [8, 9] have classified the utterances, the segments of a dialog by different speakers, into four types. These are:

- Assertion: A declarative utterance used to state facts. *Yes* and *No* in response to a question are classified as assertions on the basis that they are supplying information.

- Command: An utterance intended to instigate action. A command is generally in an imperative form, but it could be indirect such as *"My suggestion would be that you do..."*.

- Question: An utterance which is intended to elicit information, including indirect forms such as *"I was wondering whether I should..."*.

- Prompt: An utterance which does not express propositional content. *"Yeah"*, *"Okay"* and *"Uh huh..."* are examples of prompts.

Utterance Type	Controller
Assertion	speaker, unless response to a Question
Command	speaker
Question	speaker, unless response to Question or Command
Prompt	hearer

Table 2.1: Four rules for the allocation of control. These rules are referenced from Whittaker et al. [8, 9].

Based on the utterance type classification, Whittaker et al. analyzed the rules for the allocation and transfer of control. The control rules as specified in [8] and [9] are shown in Table 2.1. As an example, if the utterance is an Assertion, the controller of the dialog's initiative should be the speaker of this utterance, unless this Assertion is a response to a Question. Table 2.2, which is referenced from [8], illustrates an example dialog between participants A and B showing the shifts of control. At the beginning of the dialog, participant A first utters an Assertion. Since it is not a response to a Question, participant A (speaker) takes the initiative. After participant B replies with a Prompt, participant A also produces a Prompt. This causes a shift of control, since the initiative should be controlled by the hearer for a Prompt. Hence, the control is shifted from participant A to participant B at this stage.

Speaker	Utterance	Utterance Type	Controller
A	"And they are, you'll find that they've re-located into the labeled common area."	Assertion	A
B	"That's right."	Prompt	A
A	"Yeah."	Prompt	B
CONTROL SHIFT TO B			
B	"I've got two in there. There are two of them."	Assertion	B
A	"Right."	Prompt	B
B	"And there's another one which is % RESA"	Assertion	B
A	"OK um."	Prompt	B
B	"VS."	Assertion	B
A	"Right."	Prompt	B
B	"Mm."	Prompt	A
CONTROL SHIFT TO A			
A:	"Right and you haven't got - I assume you haven't got local labeled common with those labels."	Question	A

Table 2.2: An example dialog between A and B adapted from [8] showing the shifts of control.

2.2 Mixed-Initiative Spoken Dialog Systems

A spoken dialog system is an advanced application of spoken language technologies. It supports conversations between the human and a computer in order to carry out some tasks. Generally, spoken dialog systems can be classified into two types:

- **Transaction-based** systems allow users to execute some transactions, such as buying or selling stocks, or reserving a seat on a plane. In the interaction, both the user and the system should influence the conversations in order to complete the transaction. Hence, the mixed-initiative dialog model is suitable for the dialog management of these systems.

- **Information-provision** systems provide information in response to a query, such as requesting information of train timetable or weather. In the dialog management, the system can either present directed questions and guide the user to complete the inquiry or it can work with the user as a team to carry out the task. Hence, both the system-initiative dialog model and the mixed-initiative dialog model are suitable for the dialog management of these systems.

A spoken dialog system can serve as a test-bed for spoken language technologies. It usually involves a set of components, as depicted from the block diagram of a spoken dialog system in Figure 2.1 [2], including:

- Speech Recognition - analyzes audio speech input signal to extract linguistic units such as words or phonemes,

- Language Understanding - analyzes the meaning of user input,

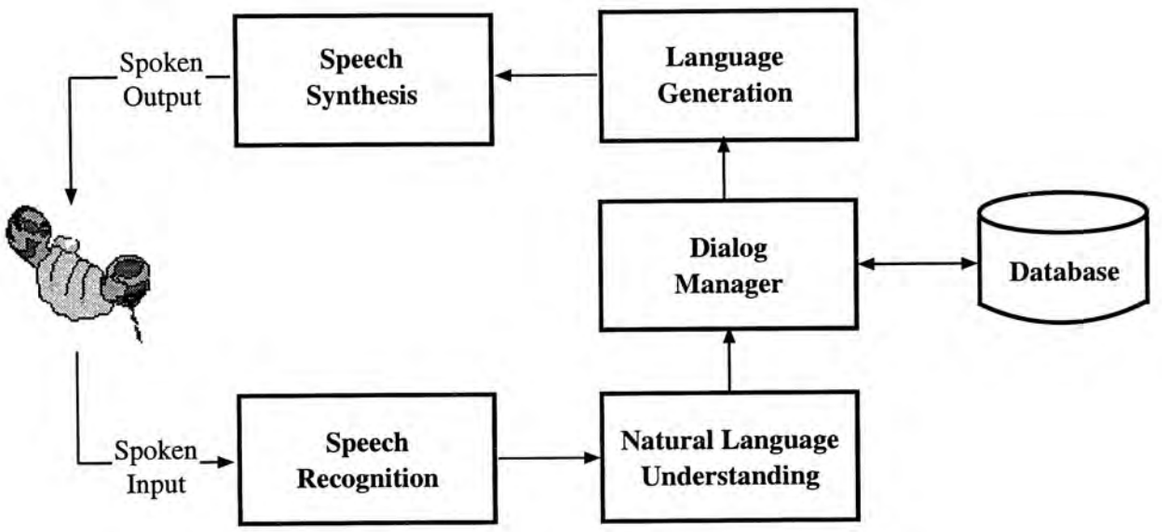

Figure 2.1: The block diagram of a spoken dialog system referenced from Glass [2].

- Dialog Management - manages the flow of the conversation, maintains history and context, accesses database and formulates responses,
- Database - stores information which provides dialog content,
- Language Generation - verbalizes responses into words, and
- Speech Synthesis - produces audio speech output signal from words.

One of the main differences in dialog systems is the degree to which the system takes an active role in the conversations. Many spoken dialog systems have been designed to support *mixed-initiative dialog model* in different restricted domains, including weather (e.g. MUXING [17] and JUPITER [18]), air travel (e.g. MERCURY [19] and PEGASUS [20]), boat traffic (e.g. WAXHOLM [21]), tourism information (e.g. LODESTAR [22]), electronic automobile classifieds (e.g. WHEELS [13]), railway services (e.g. RAILTEL [23], MASK [24] and ARISE [25]), stocks and currencies (e.g. ISIS [26] and

CUFOREX [27]) and restaurant guide (e.g. BeRP [28]). The languages concerned include English, Chinese and a number of European languages such as Swedish, Slovenian, German, Czech, Slovak, Greek, Dutch and French. Besides, some dialog systems are even multilingual and multifunctional, i.e. supporting a number of languages and a number of domain tasks instead of one single restricted domain. For example, SQEL [29] can handle four different languages and domains: German, Slovak, and Czech for the German InterCity train connections, and Slovenian for European flight information. Moreover, GALAXY [30] can handle different domains, such as automobile classified ads, restaurant guide and weather information, and different languages including English and Mandarin.

In the following sections, we will illustrate several general approaches to the dialog modeling, including the finite-state networks, form-based approaches, sequential decision approaches, reinforcement learning and the use of Belief Networks.

2.2.1 Finite-state Networks

A finite-state network [31, 32] partitions the dialog space into a finite set of dialog states and allows transitions between the states. In this paradigm, each state is associated with a topic-specific element and interaction with other system components, e.g. database. The transition arc connected with each state represents the corresponding action to be carried out. Each transition arc is usually associated with a probability value to indicate which action the system should take in the next step. Transition between states is predicted on the occurrence of specific events, either the user's inputs or

through a change in backend state.

Finite-state systems are relatively straightforward to design. Their behaviors are predictable, as the paths of dialog flow are pre-defined in the networks. However, the dialog of finite-state systems tends to be inflexible. This approach is suitable for the domains with a limited number of domain-specific concepts. However, it may become inefficient and difficult to develop when the dialog space become larger.

2.2.2 Form-based Approaches

The form-based approach [13, 33, 34, 35, 36] is an alternative dialog planning algorithm of a finite-state model. The basic data structure of the form-based approach is an electronic form (E-form), which contains a set of attribute-value slots that correspond to the required information of the application domain. For example, as referenced from [34], since the dialog manager needs to know the name of the stock and dollar amount to purchase a stock, a BUY form should contain at least two slots: STOCKNAME and AMOUNT.

The E-form usually acts as the link between the semantic frame generated by the language understanding component and the internal context maintained by the dialog management. Once the semantic frame of user's utterance is generated, the semantic information would be mapped to the canonical format of the E-form. In each dialog turn, the dialog manager would check the current status of the E-form based on a set of pre-defined rules to determine the appropriate response. For example, the dialog manager would prompt for the missing value of the key STOCKNAME when the corresponding slot is empty in the E-form. Once all the necessary slots

are filled, the dialog manager would perform the task or retrieve information from the database.

The E-form paradigm provides more flexibility than the finite-state approach does, and it can pursue a mixed-initiative dialog model. Each E-form manages requested action with one goal which states the task nature of the query or the user's intention. The E-form works well in a restricted domain. However, its format may need to be modified in order to handle requests with multiple tasks e.g. "I want to see the monthly chart of HSBC and the quotes of Cheung Kong please.". Moreover, in some task domains, the dialog may not be classified as key-value pairs easily. For example, in the restaurants domain the waiter should answer the questions regarding the restaurant's details, such as the opening and closing time, the type of food provided, the price range of the food, the location of the toilets, etc. If the system identifies the requests on the restaurant's details as one domain-specific goal, it may be difficult to classify the key-value pairs easily since this goal is vague with too much inter-related information.

2.2.3 Sequential Decision Approaches

In a sequential decision approach [3, 19, 37, 38, 39], the dialog model can be formalized in terms of a set of dialog states, actions and strategy. Dialog states characterize the amount of information available at a certain point in the dialog. Actions describe what functions the system can invoke at different dialog states. The strategy is a mapping between the dialog states and the corresponding actions. In the sequential decision process model as stated in Figure 2.2 [3], the dialog starts at the initial dialog state S_1. For

any possible states S_t the strategy prescribes what is the next action A_t to perform. Based on the interaction with the external environment (e.g. user, database, etc.), the system gets new observations O_t (e.g. output of speech recognition process, database tuples, etc.). These new observations can modify the state of the system until a final dialog state S_F is reached (e.g. the user closes the interaction).

$S_t = S_1$
while $S_t \neq S_F$ {
    $A_t = NextAction(S_t)$ ——— (*)
    Invoke A_t
    O_t = environment response to A_t
    $S_{t+1} = NextState(S_t, A_t, O_t)$
    $t = t + 1$
}

where
    S_t, A_t denote the state and action at turn t.
    S_1 and S_F stand for initial state and final state.
    Function *NextAction* determines the next action A_t.
    Function *NextState* updates the state variable.

Figure 2.2: A dialog described as a sequential process. This figure is referenced from [3].

Mixed-initiative spoken dialog systems which adopt a sequential decision approach usually require a dialog strategy for mapping an appropriate action to each dialog state (see the function *NextAction* at (*) in Figure 2.2). However, the strategy can be implemented in different ways. In the implementation of the AT&T AMICA system [3], the strategy is represented by a recursive transition network, in which the arcs represent the conditions on

the state and the nodes represent different actions. Based on this architecture, the dialog manager is required to read the current dialog state (i.e. all the information available at the current time) and use the information from the *control state* (i.e. the identification of a particular situation in the control flow of the dialog) to determine the appropriate action. In the Mercury Flight Reservation System [19], the strategy is controlled by a dialog control table. In the dialog control table, a set of ordered rules are designed which represent the mapping between the conditions of the dialog state and the corresponding action. To determine which of the actions should be performed, the system consults the dialog control table. In both systems, the rules of strategy indicated are designed by developers. Hence, the developer may need to explore all the possible situations the system might encounter.

2.2.4 Machine Learning Approaches

Machine learning [40, 41, 42] is the study of computer algorithms that can self-improve automatically and continuously through the experience and analytical observations. It always refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks may involve recognition, diagnosis, planning and prediction. Previously, some efforts have been devoted to explore the use of machine learning in automatic dialog modeling and dialog control strategies. These problems can be managed by two different approaches: reinforcement learning and the use of Belief Networks.

(i) Reinforcement Learning

Reinforcement learning [4, 43, 44] is a machine learning technique that involves an agent learning from interactions. Figure 2.3 shows the standard reinforcement learning model referenced from [4]. The learner or the decision-maker is called the *agent.* The agent interacts in the *environment.* At each time step t, the agent receives an environment state s_t. In the model of the reinforcement learning, there is a set of actions available for each state. In each interaction, the agent implements a mapping from each state representation to probabilities of selecting each possible action. This mapping is called the *agent's policy.* The agent changes its policy on the basis of its experience. According to the policy, the agent selects an appropriate action and executes it. One time step later, as a consequence of its action, the agent receives a numerical reward r_{t+1}, which causes the agent to change into a new state s_{t+1}. This process continues until the total amount of reward the agent receives over the long run is maximized.

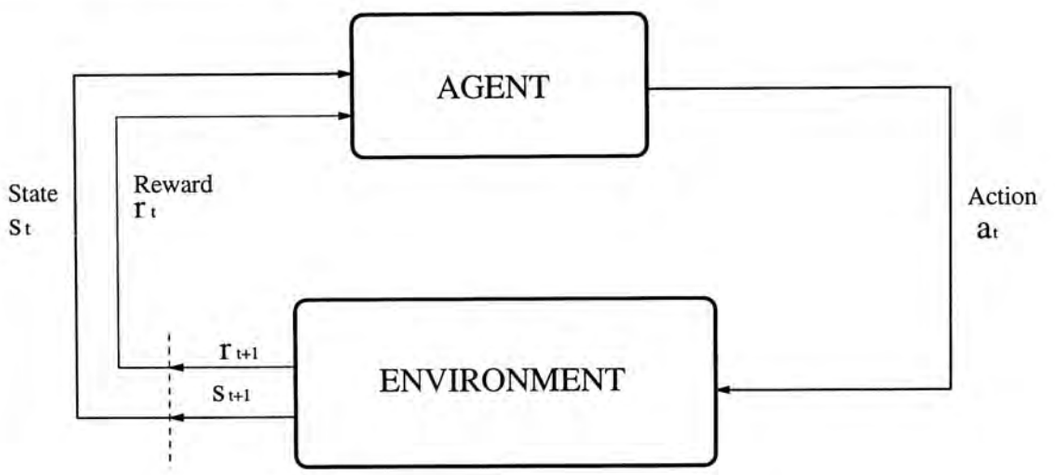

Figure 2.3: The framework of reinforcement-learning referenced from [4].

Similar to the sequential decision approach, the dialog system should

be formalized in terms of its dialog states, action set and strategy in the reinforcement learning approach. However, the agent in the reinforcement learning approach focuses on learning online during interactions with users in order to achieve its goal. Its real source of information is its own experience. The agent is not directly told which action to take, but instead it should discover which action yields the greatest reward by trying available actions. An action may affect not only the immediate reward, but also the next interaction, and consequently all subsequent rewards. These two characteristics, trial and error search and delayed reward, are the two important distinguishing characteristics of reinforcement learning.

The methodology of reinforcement learning can be applied to the automatic learning of an optimal dialog strategy [45, 46, 47], in which the dialog is formulated as an optimization problem. It can help the development of dialog systems by reducing handcrafting [48]. However, in order to learn and adapt the dialog strategy, it needs a large amount of training data. Also, selecting a good reward function may be a challenge.

(ii) Belief Networks

The Belief network (BN) [49, 50, 51, 52, 53] is a probabilistic causal network. A naïve BN topology is shown in Figure 2.4 [5] (without the dotted arrow). The black arrows indicate the causal relationships between the goal and the concepts. The naïve topology assumes that the concepts are independent from each other. In the work of [52], the topology is enhanced by means of automatic learning using the Minimum Description Length (MDL) principle [5], in which the BN not only captures the casual dependencies between the

goal and each of the concepts, but also shows the relationships among the concepts (i.e. represented by the dotted arrow).

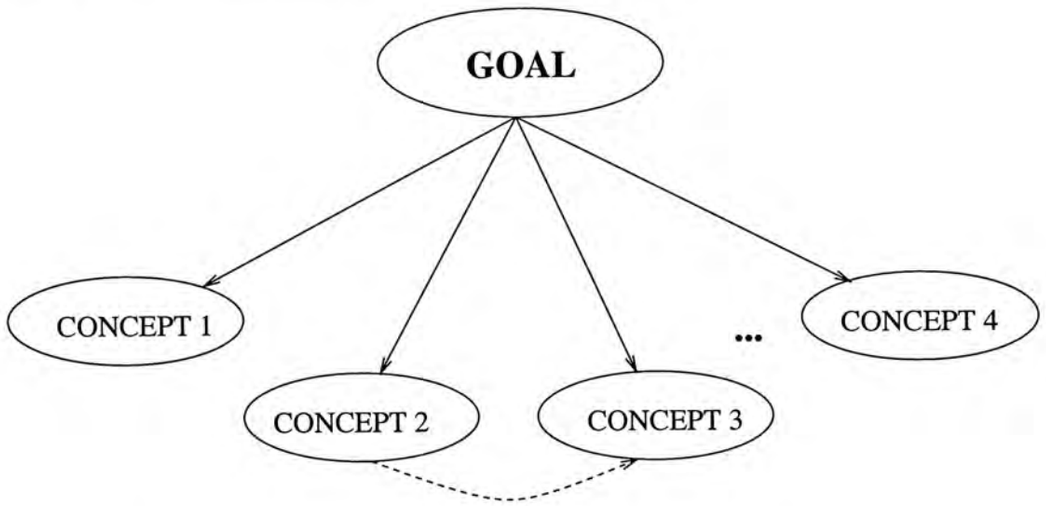

Figure 2.4: The enhanced topology of Belief Network referenced from [5].

A BN is trained for each domain-specific informational goal. The use of BN involves inferring the informational goal as well as verifying the input query against domain-specific constraints. In goal identification [54], the trained BN is used to make a binary decision based on the concepts present in the input query, regarding the presence or absence of the goal. For each query, the concepts are tagged. Based on the tagged concepts, each query is computed with an associated *aposteriori* probability by each trained BN. The probability by each trained BN is compared with a pre-set threshold. If the probability is higher than the threshold, the corresponding BN is said to vote positive to the corresponding goal. On the other hand, if the probability is lower than the threshold, then the corresponding BN is said to vote negative to the corresponding goal. The decisions across all the BNs are combined to identify the goal of an input query. The query can be labelled with the goal of the BN with the maximum probability. Alternatively, the query can

be labelled with all the goals to which the BNs vote positive. If all BNs vote negative, the input query is rejected as out-of-domain (OOD). Having inferred the informational goal of the query, the corresponding goal node is instantiated, and backward inferencing is triggered to test the networks' confidence in each input concept. Backward inferencing verifies the validity of the input query against domain-specific constraints. In this way, the system can test for cases of spurious and missing concepts in dialogs.

Moreover, BN-based dialog model can automate the development process of dialog systems. While the reinforcement learning process needs a large amount of training data to learn the optimal dialog strategy, the probabilities of BNs can be hand-designed. Hence the BN framework can still be applied to a domain even when there is a lack of training data.

In our work, we used the goal identification technology with BNs in the mixed-initiative dialog modeling in the restaurants domain. We applied the results in cooperative response generation.

2.3 Understanding Mixed-Initiative Dialogs

Mixed-initiative interaction contains many characteristics that we need to explore. Figure 2.3 shows a human-human conversation over the phone quoted from [10] where typical dialog acts are indicated on the right. In this example, human-human dialogs contain many disfluencies, interruptions, confirmations, clarification, ellipsis, co-references and sentence fragments. Since the mixed-initiative interactions are common among human, examining human-human interactions should enable us to design a mixed-initiative dialog model in a more principled way.

C:	"Yeah, [umm] I'm looking for the Buford Cinema"	*disfluency*
A:	"OK, and you want to know what's showing there or..."	*interruption*
C:	"Yes, please."	*confirmation*
A:	"Are you looking for a particular movie?"	
C:	"[umm] What's showing."	*clarification*
A:	"OK, one moment."	*back channel*
A:	"They're showing A Troll In Central Park."	
C:	"No."	*inference*
A:	"Frankenstein."	*ellipsis*
C:	"What time is that on?"	*co-reference*
A:	"Seven twenty and nine fifty"	
C:	"OK, and the others?"	*fragment*
A:	"Little Giant"	
C:	"No."	
A:	"..."	
C:	"..."	
A:	"That's it."	
C:	"Thank you."	
A:	"Thanks for calling Movies Now."	

Table 2.3: Transcription of a conversation between an agent(A) and a client (C) over the phone referenced from [10].

Recently, significant efforts have been devoted towards understanding mixed-initiative structures in human-human dialogs in comparison with human-computer dialogs [55, 56, 57]. In the study by MITRE [55], each dialog is tagged for initiative and dialog acts. Initiative tagging tags each utterance with the four rules shown in Table 2.1. Dialog act tagging uses a subset of CSTAR Consortium tags [58] for each utterance. From the evaluation, the distribution of system-initiative utterances and user-initiative utterances is even in human-human dialogs while the initiative is appeared to skew towards the system in human-computer dialogs. In addition, users ask more questions and action requests in human-human dialogs while both participants confirm agreement less frequently in human-computer dialogs.

2.4 Cooperative Response Generation

Spoken dialog systems should be capable of generating coherent responses. Whenever we ask a question, we expect a cooperative answer in the system's reply. In order to build a cooperative dialog system, it is important to determine what a cooperative answer is. Grice [59, 60, 61] proposes a number of *maxims* that characterize cooperative answers. By the *maxim of quality*, speakers are expected to respond true statements with adequate evidences. By the *maxim of quantity*, speakers are expected to give enough information needed. By the *maxim of relevance*, the responses should be relevant to the ongoing conversation. By the *maxim of manner*, the responses should be brief and orderly, to avoid obscurity or ambiguity. Despite their vagueness, these maxims can give us some ideas on what characteristics the cooperative answers should have.

In typical human-computer interaction, the user and the system do not interact perfectly. When preferences and restrictions of the user cannot be all satisfied, over-constrained situations occur. Cooperative response generation should be important at this point of dialog. In the following sections, we will illustrate two approaches for cooperative response generation, including plan-based approach [62, 63] and constraint-based approach [64].

2.4.1 Plan-based Approach

Chu-Carroll and Carberry [63] presented a plan-based architecture that captures the *Propose-Evaluate-Modify* cycle of collaboration for response generation, with emphasis on cases in which the system and the user disagree. Conflicts may arise between conversants during a dialog. In order to be collaborative, a response generation system should be able to detect those conflicts as soon as they arise, and to engage in negotiation with the user to resolve the conflicts. In the *Propose-Evaluate-Modify* framework, the user proposes a set of actions and beliefs to be incorporated into the shared plan being developed. Then the system evaluates the proposal based on its knowledge and determines whether or not it accepts the proposal. If not, the system would propose a set of modifications to the user's original proposal. These modifications would become the system's proposal to the user. The user evaluates it and, if conflicts arise, he may propose modifications to the system's proposal, resulting in a recursive process. This model can facilitate the recognition of user goals as the basis for a response generation system. However, it only focuses on the evaluation and modification of proposed invalid beliefs and dislikes.

2.4.2 Constraint-based Approach

Over-constrained situations occur when the preferences and restrictions of the user cannot be totally satisfied. Qu and Beale [64] presented a constraint-based model for cooperative response generation, with an emphasis on detecting and resolving over-constrained situations by using a constraint satisfaction problem (CSP) framework. The CSP framework consists of cycles of *constraint acquisition*, *solution construction*, *solution evaluation* and *solution modification*. It attempts to find consistent assignment of values to a fixed set of attributes of the user's information needs. For example, in the travel domain, attributes such as *arrival-city*, *departure-city*, *date*, *carrier* and *time* are possible variables. Each of these attributes should be associated with some constraints, which are controlled by domain relations in the database or user preferences and restrictions. Besides, each attribute should be associated with different strengths of constraints, e.g. the departure city and the arrival city are usually *required* while the carrier information is *preferred* but not required. In the *constraint acquisition* phase, the system gathers constraints through recognizing and requesting constraints from the user input, and proposing constraints for the user to confirm. In the *solution construction* phase, all solutions to the CSP are generated iteratively by combining partial answers into a complete list of correct answers. During the *solution evaluation*, the system analyzes the generated answers and detects the over-constrained and under-constrained situations. If no solution can be found to satisfy the user's information needs, the system provides relaxed solutions and informs the user of the over-constrained situation. Then the *solution modification* module is invoked for the over-constrained situations.

However, if the system modifies the problem definition without the user's consent, it would be considered uncooperative. Thus, the system goes back to the constraint acquisition phase to interact with the user, resulting in a recursive cycle until a satisfactory solution is found. In comparison, the CSP-based framework can formulate and handle more types of cooperative responses, while the plan-based approach only focuses on the responses for user's dislike situations.

2.5 Chapter Summary

In this chapter, we have presented some previous work in the areas of the mixed-initiative dialog modeling, spoken dialog systems and cooperative response generation. First, we have given a brief introduction and design challenges of mixed-initative interactions. Then, we have presented several approaches to mixed-initiative dialog modeling in existing spoken dialog systems. Also, we have introduced some recent studies of mixed-initiative structures and comparisons between human-human and human-computer dialogs. Lastly, we have described some key points and approaches for handling cooperative response generation. In the next three chapters, we will describe our work in mixed-initiative dialog modeling in detail.

Chapter 3

Mixed-Initiative Dialog Management in the ISIS system

In this chapter, we present our work on mixed-initiative dialog management in a spoken dialog system called ISIS. ISIS, which abbreviates as Intelligent Speech for Information System, is a trilingual spoken dialog system for the stocks domain supporting English, Cantonese and Putonghua. Within the context of ISIS, we have designed and implemented a mixed-initiative dialog model capable of (i) inheriting selected discourse history, (ii) combining online interaction with offline delegation sub-dialogs, as well as (iii) learning newly listed stocks to expand the system's knowledge base through an interaction with the user. In ISIS, discourse inheritance is achieved by using a form-based approach. The response generation is controlled by the templates which state the mapping between the query cases and their corresponding response trigger words. In the dialog model of ISIS, the shifting of initiatives only depends on the task goal.

3.1 The ISIS Domain

3.1.1 System Overview

ISIS [26, 6, 65, 66], which abbreviates as *Intelligent Speech for Information System*, is a spoken dialog system for the stocks domain. It is a trilingual system supporting English, Cantonese and Putonghua - the predominant languages used in our region, as well as multi-modal and mixed-modal input. Input modalities include speaking, typing and mouse-clicking. Output media include synthesized speech, text, tables and graphics. The system resembles a virtual stock broker, which can provide the user with real-time stock market information and personal portfolio information. It also handles simulated financial transactions. The financial domain is of particular interest to our region, which is one of the world's financial centers.

Figure 3.1 illustrates the ISIS system architecture referenced from [6]. The client supports both the text I/O and audio I/O through the applet. There are six server objects in the ISIS framework, including speech recognition, language understanding, speech generation, speaker authentication, dialog manager, and the time-out manager. The server objects and the browser-based client objects communicate via the intranet or Internet with the IIOP (Internet InterORB Protocol) supported in CORBA architecture. Additionally, there is a pair of KQML (Knowledge Query and Manipulation Language) software agents that communicate via the facilitator to handle the user's price alert requests. Among all the server and client objects, data is passed through the format of XML (EXtensible Markup Language)[1]. The

[1]http://www.w3.org/XML

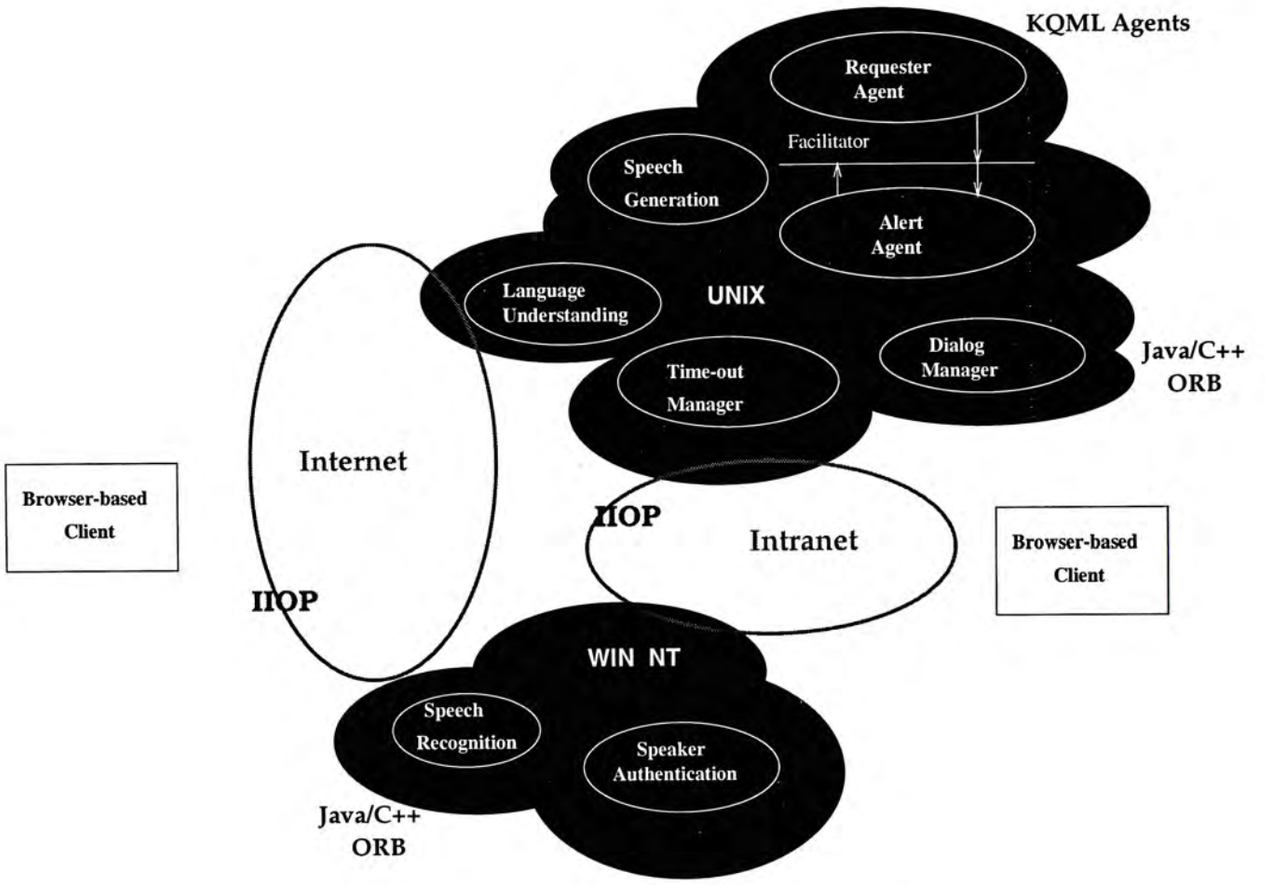

Figure 3.1: Overview of the ISIS System Architecture. This figure is referenced from [6].

data is labeled with the descriptive semantic tags characterizing the server class operations. Figure 3.2 shows the XML message generated by language understanding (LU) component based on the user input *"buy five lots of HSBC at the market price please."*, where "ric" denotes Reuters Instrument Code and "iv" denotes in-vocabulary.

```
<LU>
    <BUY>
      <stock type=ric status=iv>0005.HK</stock>
      <price type=equal>market</price>
      <lot type=equal>5</lot>
    </BUY>
</LU>
```

Figure 3.2: An example XML message produced by the Language Understanding (LU) server class. Input query was *"buy five lots of HSBC at the market price please."*. In the XML message, "ric" denotes Reuters Instrument Code and "iv" denotes "in-vocabulary".

3.1.2 Domain-Specific Constraints

Domain-specific constraints define the task domain and support the mixed-initiative control in ISIS. Prior to the development of ISIS, we have identified *ten* domain-specific goals for the stocks domain, which includes real-time quotes, news, order-amendments, buy, sell, portfolio inquiry, etc. To aid the task of domain definition, we collected some sample queries in both English and Chinese. We have requested our subjects to compose questions that they may ask of a stock broker, e.g. questions on the real-time stock quotes or simulated investor accounts. Based on the collected requests for

each domain-specific goal, we compiled a set of concepts deemed *related* to the goal. We also referenced the relational database for additional related concepts. In this way, we defined the set of domain-specific constraints in the ISIS domain, i.e. each goal has its own set of related concepts. Table 3.1 illustrates these ten goals with the corresponding domain-specific constraints. Similar to Constraint Satisfaction Problem (CSP) approach (see Section 2.4.2 in Chapter 2), the system receives the user's query and checks whether the query has provided information which satisfies the pre-defined constraints. For example, an inquiry regarding *BUY* transaction mandates that a stock code, a bid price value and a number of lots or shares should be specified. Besides, each attribute is associated with different strengths of constraints, e.g. the action value and the stock code are always *required* while the time value, the number of lots or share and the price value are *preferred* but not required.

In the flow of ISIS, the Language Understanding (LU) [67] module parses the user's queries and infers the corresponding domain-specific goal(s). Regarding the queries which do not belong to these ten pre-defined domain-specific goals, LU treats them as out-of-domain (OOD) queries. After the process of understanding, LU sends the result to the Dialog Manager via an XML message. An example XML message produced by LU is shown in Figure 3.2.

3.2 Discourse and Dialog

The Dialog Manager (DM, see Figure 3.1 in Section 3.1.1) manages the dialog flow in ISIS. Dialog control is distributed as each object (server / client)

Real-Time Quotes
stock code/name e.g. 0001.HK, 0005.HK
News
stock code/name e.g. 0001.HK, 0005.HK, and *time e.g. yesterday, today*
Chart
stock code/name e.g. 0001.HK, 0005.HK, and *period e.g. daily, weekly, monthly*
Trends
stock code/name e.g. 0001.HK, 0005.HK, and *time e.g. yesterday, today*, or *direction e.g. up or down*
Buy
stock code/name e.g. 0001.HK, 0005.HK, and *price e.g. ninety six dollars and fifty cents*, and *number of lots or shares e.g. five lots, four hundred shares*
Sell
stock code/name e.g. 0001.HK, 0005.HK, and *price e.g. ninety six dollars and fifty cents*, and *number of lots or shares e.g. five lots, four hundred shares*
Order-Cancel
action e.g. buy, sell, and *stock code/name e.g. 0001.HK, 0005.HK*, or *time e.g. yesterday, today*, or *number of lots or shares e.g. five lots, four hundred shares*, or *price e.g. ninety six dollars and fifty cents*
Order-Amendments
old action e.g. buy, sell, and *new action e.g. buy, sell*, and *stock code/name e.g. 0001.HK, 0005.HK*, or *price e.g. ninety six dollars and fifty cents*, or *number of lots or shares e.g. five lots, four hundred shares*
Portfolio Inquiry
stock code/name e.g. Cheung Kong, 0001.HK, and *action e.g. buy, sell*
Meta Command
command e.g. where, bye, yes, no, refresh, undo

Table 3.1: Ten domain-specific goals and the corresponding required attributes in ISIS domain. For example, an inquiry regarding *BUY* transaction mandates that a *stock code* or a *stock name*, a *price* value and a *number of lots or shares* should be specified.

which keeps track of its successor object(s) in the processing pipeline. In the process flow of ISIS, DM is the successor of the Alert Agent object, the Language Understanding (LU) object and the Time-out Manager (TM) object. TM monitors the time between successive user's inputs. If the time duration exceeds a pre-set threshold (i.e. the user has been silent for a while), TM sends a message to the DM object. The Alert Agent object monitors the real-time financial data feed continuously and initiates an alert message when the user's pre-specified price is met; thus the offline delegation sub-dialog is invoked. LU interprets the user's query from words to intentions and sends the results to DM. Hence DM processes the messages from its three predecessor objects differently:

- If the message is received from TM, DM invokes its response generation procedure to produce the system response, *"Are you there?"* and then repeats the last system response.

- If the message is received from the Alert Agent, DM handles it as an offline delegation sub-dialog, which will be described in Section 3.3.2.

- If the message is received from LU, DM invokes a series of procedures / steps: (1) checking for missing attributes; (2) discourse inheritance; (3) validation and detection of specific variables and tags; (4) information/database access; (5) validation and updating the variables of E-forms; and (6) response frame generation.

The successor of DM is the Speech Generation (SG) server object. SG can invoke various speech synthesizers. After finishing the process of dialog management, DM sends an XML response frame to SG and hence the user can

hear the verbalized responses from the system.

In our dialog model, ISIS can inherit information from the past queries, prompt for missing information, invoke confirmation at critical stages of dialogs, handle out-of-domain queries, provide meta commands, offer price alert services, and generate the time-out messages. Table 3.2 shows seven example dialog turns of ISIS. In the next two subsections, we will illustrate both the computation and representations used in discourse inheritance and mixed-initiative dialog modeling in detail.

3.2.1 Discourse Inheritance

Discourse inheritance inherits past values from discourse context to the current user input. In ISIS, discourse inheritance is achieved by using an electronic form (E-form) model [13], which is a semantic frame containing many key-value slot pairs for the required pieces of information. Figure 3.3 illustrated our modified E-form model regarding a user query *"Show me the quotes of Cheung Kong and HSBC please"*. In this E-form, the field *Number_of_goals* indicates the number of goals interpreted from the user query. Here, the user has requested two actions in the same query:

Show me the quotes of Cheung Kong and

show me the quotes of HSBC.

Hence the *Number_of_goals* is 2 in the example. In ISIS, the dialog model supports E-forms from both mixed-initiative online interaction (OI) and offline delegation (OD). The field *Inheritance_indicator* marks whether this E-form comes from OI or OD which will be described in Section 3.3.2 in detail. Furthermore, the field *Records* stores the action goals and attributes

ISIS	:	"Welcome to ISIS. Please input your query."
Dialog 1:	:	Handling out-of-domain queries.
User1	:	*"I would like the exchange rate of the Singapore dollar please."*
ISIS	:	"Sorry, I don't understand your request. I can provide only information about stocks such as real-time quotes, news and charts. I can also provide information about the simulated personal portfolio, and handle buy and sell transactions."
Dialog 2:	:	Prompting for the missing information.
User	:	*"I would like to buy three lots of Cheung Kong Holdings please."*
ISIS	:	"Please provide the bid price."
Dialog 3:	:	Inheriting information from the past queries
User	:	"Can you show me the news of today first."
ISIS	:	"The news of 0001 (a stock code) Cheung Kong Holdings Ltd is shown below. "(news shown).
Dialog 4:	:	Invoking confirmation at the critical stages of dialogs
User	:	*"Buy three lots at eighty nine dollars please."*
ISIS	:	"Please say the following digits in English to verify your identity: 1 3 7 6 4." (Speaker verification is triggered for transaction process)
User	:	*"One, three, seven, six, four."*
ISIS	:	"Please say the following action: buy three lots of Cheung Kong Holdings Limited at eighty-nine dollars. Please say yes to confirm or no to reject to action."
User	:	*"Yes"*
Dialog 5:	:	Offering price alert services
ISIS	:	"The current bid price for Cheung Kong Holdings is ninety dollars and ten cents. Do you want us to notify you when the price drops to eighty nine dollars? Please say yes or no."
User	:	*"Yes"*
ISIS	:	"The Alert Agent is launched!"
User	:	*"Show me the monthly chart of HSBC."*
ISIS	:	"The monthly chart of HSBC is shown below."(graphic shown)
ISIS	:	"There is one alert message for you. The stock price of Cheung Kong Holdings Limited is at eighty-nine dollars presently. Please confirm your previous request on May 30, 2002 at 2:59pm. You would like to buy three lots of Cheung Kong Holdings Limited at your requested price of eighty nine dollars per share. Please say yes to confirm or no to reject the action."
User	:	*"Yes"*
ISIS	:	"The buy transaction is completed. Please input a new query."
Dialog 6:	:	Generating the time-out messages
(15 minutes later)		
ISIS	:	"Are you there? Hello?"
Dialog 7:	:	Handling the meta commands
User	:	*"Bye."*
ISIS	:	"Goodbye and see you later."

Table 3.2: Seven example dialog turns from the ISIS system.

of the corresponding user input.

Number_of_goals	2
Inheritance_indicator	from_online_interaction
Records	Goal: REAL_TIME QUOTES; Stock code: 0005.HK Goal: REAL_TIME QUOTES; Stock code: 0001.HK

Figure 3.3: An example E-form of the user query *"Show me the quotes of Cheung Kong and HSBC please"* with the attributes *Number_of_goals*, *Inheritance_indicator* and *Records*.

$User_{t-1}$	: *"I would like to check the quotes of HSBC."*
$System_{t-1}$	: "Here is the real-time quotes of HSBC." (quotes shown)
$User_t$	: *"How about the monthly chart?"* (missing stock code)
$System_t$	: "The monthly chart of **HSBC** is shown below." (graphic shown)

Table 3.3: An example dialog illustrating discourse inheritance.

In every dialog turn, ISIS checks for the *missing* attributes and inherits the values *selectively* from the discourse to the current user input. Domain-specific constraints (see Table 3.1 in Section 3.1.2) help to check for missing attributes from the current input. Table 3.3 shows an example dialog illustrating discourse inheritance from ISIS. The user first requests to check the real-time quotes of HSBC at time *t-1*. The system replies with the quotes shown in the interface. The user requests again by saying *"How about the monthly chart?"*. With the help of domain-specific constraints, DM checks

the information provided by the user and detects the missing attributes of the goal stated in the message. In the example, DM detects that the goal *CHART* requires two pieces of information including a stock code and a chart type as stated in the constraint table Table 3.1. Hence it indicates the status "missing" for the attribute *stock code* in the E-form. Figure 3.4 depicts the modified version after detecting the missing attribute "stock code". In the next step, DM attempts to inherit the latest past value from discourse context to the missing attribute of the user input *selectively.* From the discourse context, DM detects that the latest past value of the stock code is "0005.HK" of the stock "HSBC" at time *t-1.* Hence, DM updates the value of the attribute *stock code* from "missing" to "0005.HK" for the current user input, as shown in Figure 3.5. With all the information available for the action *CHART*, the system can execute the function by retrieving the monthly chart of HSBC from the database and passes the graphic to the user.

Number_of_goals	1	
Inheritance_indicator	from_online_interaction	
Records	Goal	CHART
	Chart Type	monthly
	Stock code	**missing**

Figure 3.4: A modified E-form after the checking for missing stock code.

In discourse inheritance, semantic concepts from the current user's query take precedence over previous query (queries) from LU. DM manages the online interaction discourse history in a list with a time sequence as shown in Figure 3.6. Each E-form has an inheritance indicator that indicates whether

Number_of_goals	1	
Inheritance_indicator	from_online_interaction	
Records	Goal	CHART
	Chart Type	monthly
	Stock code	**0005.HK**

Figure 3.5: A modified E-form after discourse inheritance, continuing from Figure 3.4.

the corresponding dialog state belongs to online interaction or offline delegation. E-forms which belong to offline delegation have no information related to the previous online interaction. Hence in discourse inheritance the dialog state which belongs to online interaction cannot inherit value from offline delegation and vice versa.

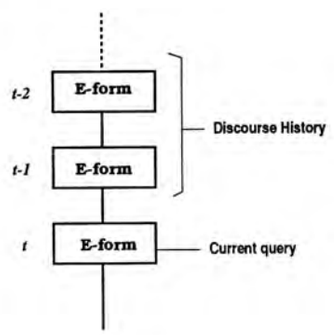

Figure 3.6: Online interaction discourse history is managed in a list with a time sequence.

3.2.2 Mixed-Initiative Dialogs

In ISIS, a mixed-initiative dialog model is used. In the dialog flow, DM processes the received messages from TM, the Alert Agents and LU. Ac-

cording to the rules for the allocation of control referenced from [8] and [9], the initiative only shifts to the system under *three* query conditions in ISIS, including:

- **Case 1: Prompting for missing information**

  If the specific attribute of the goal stated in the domain-specific constraints is absent in the input query, a missing attribute is detected. Hence the system will prompt for the missing concept. Table 3.4 shows an example dialog with the query "Buy three lots of Cheung Kong please.". In the domain-specific constraints (see Table 3.1), an inquiry regarding *BUY* transaction mandates that a stock code, a bid price value and a number of lots or shares should be specified. As a result, the attribute "price" is missing in the query, and the system generate a prompt "Please provide the bid price.".

User : *"Buy three lots of Cheung Kong please."*
ISIS : *"Please provide the bid price."*

Table 3.4: An example dialog interaction for the query case of the missing attribute "price".

- **Case 2: Invoking confirmation sub-dialogs**

  Systems should execute the requested actions whenever all the information is ready. In ISIS, if the goal is related to the user's portfolios, i.e. BUY, SELL, ORDER-CANCEL, ORDER-AMENDMENTS and PORTFOLIO INQUIRY, the system prompts for a confirmation first instead of executing the action. On the other hand, if the action is not related to the user's portfolio, i.e. REAL-TIME QUOTES, NEWS, CHART and TRENDS, the system executes the action directly and

tells user the results without invoking any confirmation sub-dialog. Table 3.5 shows an example for invoking a confirmation sub-dialog. After the user requests "Buy three lots of Cheung Kong at the market price please", DM detects that the information for executing the BUY action is available and hence invokes the speaker verification (SV) process to identify the user's identity. Since the action BUY can affect the user's asset, we need the user's confirmation for executing the action. As a result, the system prompts for a confirmation "Please confirm the following action. Buy six lots of HSBC at ninety nine dollars. Please say yes to confirm or no to reject the action.".

User : "*Buy three lots of Cheung Kong at the market price please.*"
(System invokes SV for the buy transaction)
ISIS : "*Please confirm the following action. Buy six lots of HSBC at ninety nine dollars. Please say yes to confirm or no to reject the action.*"
User : "*Yes.*"

Table 3.5: An example of the confirmation sub-dialogs for the user's portfolio action "*BUY*".

- **Case 3: Offering price alert services**

  Users of ISIS can delegate tasks to the Alert Agent. In ISIS, the Alert Agent object is responsible for monitoring the real-time financial data feed continuously and initiating an alert message when the user's pre-specified price is met, thus the offline delegation sub-dialog is invoked. The typical request is to monitor a specified stock price hitting a particular price point, e.g. "Please alert me when the bid price of Cheung Kong is eighty nine dollars". Alternatively, a buy or sell request for which there is a mismatch between the ask/bid price and the market

price will also cause an agent to be launched as shown in Table 3.6.

User :	*"Buy three lots at eight nine dollars please"*
(System invokes SV for the buy transaction)	
ISIS :	*"Please confirm the following action: buy three lots of Cheung Kong Holdings Limited at eight-nine dollars. Please say yes to confirm or no to reject to action."*
User :	*"Yes."*
ISIS :	*"The current bid price for Cheung Kong Holdings is at ninety dollars and ten cents. Do you want us to notify you when the price drops to eighty nine dollars? Please say yes or no."*
User :	*"Yes."*
ISIS :	*"The Alert Agent is launched!(Time stamp for the agent is May 30 2002 at 2:59 PM)"*
User :	*"Show me the monthly chart of HSBC."*
ISIS :	*"The monthly chart of HSBC is shown below. (graphic shown) There is one alert message for you. The stock price of Cheung Kong Holdings Limited is at eighty nine dollars presently. Please confirm your previous request on May 30, 2002 at 2:59PM. You would like to buy three lots of Cheung Kong Holdings Limited at your requested price of eighty nine dollars per share. Please say yes to confirm or no to reject the action. (The Notifier agent came back with an alert message.)"*
User :	*"Yes."*
ISIS :	*"The buy transaction is completed. Please input a new query."*

Table 3.6: An example dialog illustrating price alert services.

DM can check the query cases detected by LU, the Alert Agent and TM from the received messages. In the dialog flow, each query case is mapped to a unique response trigger word, which is a key flag to invoke the appropriate text or spoken response to the user. The mapping of turn management is summarized in Table 3.7. As an example, DM can get the query cases detected by LU including: meta-commands, OOD queries, missing attributes, inappropriate confirmations, e.g. the user suddenly says "yes" after the confirmation subdialogs, and invalid data retrieval, e.g. the requested number

of shares is *not* a multiple of the lot size and the bid price value is lower than the market price.

Query Cases		Response Trigger
(from LU)	Meta-commands	meta-command
	OOD queries	reject
	Missing attribute	missing
	Inappropriate confirmations	reject
	Invalid data retrieved	invalid
(Default)	User's communicative goal	goal dependent (e.g. confirmation, success)
(from Alert)	Possible user alerts	alert message
(from TM)	Time-out cases	time-out

Table 3.7: A summary for turn management. "LU" denotes the Language Understanding component while "Alert" indicates the Alert Agent object and "TM" denotes the Timeout Manager object.

3.3 Challenges and New Directions

ISIS works in the financial domain which presents two complexities for dialog management in a spoken dialog system, including the newly listed stocks problem and the interaction involving multiple tasks. In the stocks domain, new companies continue to be listed, which requires that our system is extensible to accommodate the new stock listings. In addition, in real situations the stockbrokers always provide multiple services simultaneously. Each day, the stockbrokers have to monitor the stock, interact with the client and provide an alert service to notify the client when a specific stock price hits a particular price point. ISIS aims to resemble a virtual stock broker. While providing multiple tasks on dialog management as a real stock broker, the system should be able to handle the sudden alert messages received during

online interaction with the user. Hence, our system should be able to transit between online interaction and offline delegation. We will illustrate our work on these two complexities [6, 65] in the following subsections in detail.

3.3.1 A Learning System

We started to develop ISIS into a learning system that can automatically incorporate the newly listed stocks into its knowledge base through a text-based conversation. This is a desirable feature for our application domain because new stocks are continually added to the listing at the stock exchange. For example, the Mass Transit Railway Corporation was listed at the Stock Exchange of Hong Kong during the time of the research, under the name "MTR Corporation" according to our dedicated Reuters feed. This company is commonly referred to as "MTR" in Hong Kong. Since the listing is new, none of these names exist in the ISIS knowledge base.

We start to tackle the newly listed stocks problem at the natural language level [68]. At the moment we only apply this work on the user inputs that are text-based, hence bypassing the issue that have to do with misrecognition of out-of-vocabulary (OOV) words, in order to focus on the dialog aspects. The automatic learning process begins when a user types in a query with an out-of-vocabulary (OOV) stock name such as *"Do you have the real-time quotes of MTR?"*. Then, the LU server object understands and employs a transformation-based parsing technique [69] to interpret the word *MTR* as an OOV stock name. DM receives the result from LU containing an XML tag <stock type=name status=oov> and invokes a series of procedures:

- check an XML tag <stock type=name status=oov>,

- invoke a substring matching to find the corresponding listing,
- invoke a confirmation sub-dialog which elicits new grammar rules from user: *stock_name* $\rightarrow$ *MTR*, *stock_code* $\rightarrow$ 0066.*HK*, and
- call LU server to add grammar rules

An example dialog for automatically incorporating the newly listed stock MTR into our knowledge domain is shown in Table 3.8. The automatic learning process begins when a user inputs a query *"Do you have the real-time quotes of MTR?"*. Since we bypass the problem of recognizing of OOV words, the word MTR is not known to the system. Then, based on the transformation-based parsing technique [69], the LU server object identifies that the OOV word MTR is a member of the semantic class **stock** and tags it as <stock type=name status=oov>. Hence in this task, a totally unknown word for the semantic class **stock** could be added. The augmentations are not restricted to subsets of the existing names.

Referring to the series of procedures in DM (see Section 3.2), DM detects the <stock type=name status=oov> tag and invokes a procedure that probes Reuters' name listings with a substring matching algorithm. The matching list with better matches ranked higher is sent to the client object. The system asks for the correct stock code for "MTR". The user follows the prompt and states the stock code "zero zero six six" to the system. DM interprets the spoken code correctly by the context inheritance. At this moment, the system realizes that the user wants to know the real-time quotes of the stock with the code "0066" and the listed name "MTR Corporation" from the Reuters' database. Hence, DM retrieves the real-time quotes of the

User1 :	*"Do you have the real-time quotes of MTR?"*
LU produces the E-form in XML format: <QUOTES> <stock type=name status=oov>MTR</stock> </QUOTES>. DM detects the <stock type=name status=oov> tag in the SQL generation step, and invokes a procedure that probes Reuters' name listings with a substring matching algorithm. Better matches are ranked higher in the displayed list.	
ISIS1 :	*"I do not know of MTR. This name is similar to the following list of stocks, together with their stock codes. Please speak the stock code for correct choice. Otherwise please ask for another stock."* 0066: MTR Corporation 1005: MATRIX Holdings 0375: YGM Trading
User2 :	*"It should be zero zero six six."*
DM interprets the spoken code correctly by the context inheritance. The following subdialog attempts to match the commonly used stock name with the listed name.	
System2 :	*"Here are the real-time quotes for MTR. (Quotes shown) Do you generally refer to MTR CORPORATION as MTR? Please say yes or no."*
User3 :	*"Yes."*
Upon user confirmation, DM calls LU to add the rules stock_name → MTR and stock_code → 0066 to its grammar.	
System3 :	*"I got it. How else may I help you?"*
User 4 :	*"Show me the latest new of MTR please."*
LU's new grammar rule is now capable of interpreting queries regarding the newly listed company.	
System4 :	*"There is no news for MTR today."*

Table 3.8: An example dialog showing the automatic incorporation of new stock MTR into the ISIS knowledge base.

stock from Reuters' data feed. In the next step, ISIS attempts to match the commonly used stock name with the listed name by asking *"Do you generally refer to MTR Corporation as MTR? Please say yes or no"*. If the user confirms the matching by saying "Yes", it means that the user always refers to MTR Corporation as MTR. DM then calls LU to add the rules *stock_name* $\rightarrow$ *MTR* and *stock_code* $\rightarrow$ 0066 to its grammar. After LU successfully adds the rules, it sends a "success" flag to DM. The system then replies to the user by saying *"I got it"*. Hence from now on, LU's new grammar rule is capable of interpreting queries regarding the newly listed company MTR. If the user says any query regarding MTR such as "Show me the latest news of MTR please", the system can understand the query since the stock name *MTR* is already in our domain. Hence in the example the system replies *"There is no news for MTR today"* to the user since it accesses the database and finds no news for MTR today.

In the case of Chinese, OOVs are identified by an n-gram grouping technique together with transformation-based parsing [69]. Subsequent operations are the same as those for English.

3.3.2 Combining Interaction and Delegation Subdialogs

ISIS can delegate tasks to a pair of software agents implemented in KQML. KQML is both a message format and a message-handling protocol to support information exchanges among agents. A non-blocking request from the user query is sent to the Requester Agent, which communicates the message to the Alert Agent through the Facilitator, as shown in Figure 3.7. A typical request for the Alert Agent is to monitor a specified stock price hitting a

particular price point. Users may launch an agent with a request e.g. *"Please notify me when Cheung Kong Holdings rises to ninety two dollars per share"*. Alternatively, a buy or sell request for which there is a mismatch between the ask/bid price and the market price will also cause an agent to be implicitly launched. When the specified condition is met, the Alert Agent will send an alert message through the Facilitator and the Requester Agent and back to the user. Since the alert message may arrive back to the user during the interaction, ISIS should allow a transition between a mixed-initiative online interaction (OI) and an offline delegation (OD) in a single dialog thread, which controls the flow of interaction with the user. In ISIS, task delegation and the ability to switch between OI and OD during interactive conversation with the user supports both the spoken and text-based input/output modes.

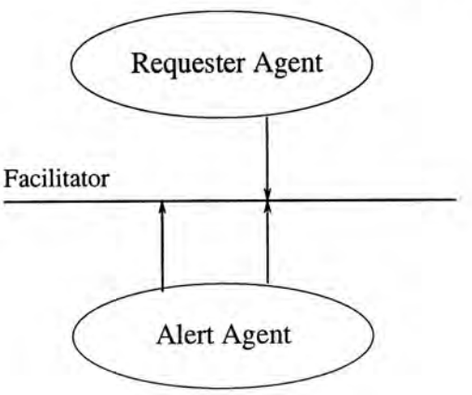

Figure 3.7: A pair of software agents in ISIS. A non-blocking request from the user query is sent to the Requester Agent, which communicates the message to the Alert Agent through the Facilitator.

In our work, we have taken a first step to investigate how a system can manage dialogs regarding the switching between online interactions and offline delegations. The following is admittedly a preliminary solution, and much more still needs to be done. In ISIS, the transition is achieved by main-

taining two lists of E-forms namely OI list and OD list. The OI list stores the online interaction discourse history, in which the system and the user take alternate turns in the mixed-initiative interactions. The OD list stores a sequence of alert messages received from the Alert Agent. We believe that each message should have a specific referential anchor for the user. In our implementation, we have chosen to use the *stock name* as the anchor word of alert messages. Moreover, there is a register which is responsible for storing the dialog state from IO before switching to OD. While handling the transition, the system would attach the E-forms from OD list to OI list since OI list controls the online interactions. However, discourse inheritance should not mix up the E-forms from OD list with those belonging to online interaction in OI list. In our implementation, we use the field "Inheritance_indicator" in the E-forms to distinguish where they are come from. Figure 3.8 shows the initial state before the dialog involving multiple tasks.

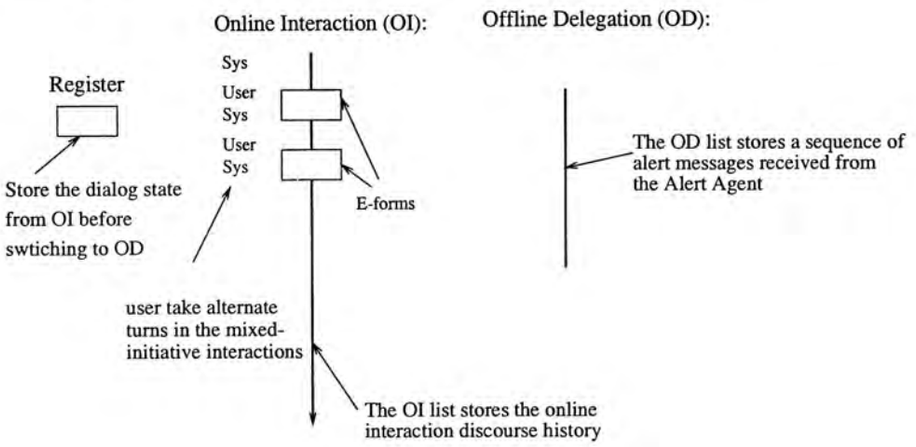

Figure 3.8: Initial state before the dialog involving multiple tasks. In ISIS, the transition is achieved by maintaining two lists of E-forms namely OI list and OD list, and a register.

We first explain the mechanism for handling the transition between OI

and OD. If DM suddenly receives an alert message from the alert agent during the online interactions, DM first attaches the incoming alert message into the OD list. Then, DM checks the current dialog state in online interaction. If the OI dialog is at a stage right before the transaction completion, e.g. confirming a buy transaction, DM does not allow an interruption by the sudden alert. However, if the dialog state is at the state of completion,[2] DM offers the user two options: he can either continue in the current synchronous dialog and ignore the message, or he can store the current synchronous dialog and switch to handle the asynchronous alert. At this time, DM sets a global variable $\alpha = 1$ and indicates the possibility of switching from online interaction to offline delegation. Figure 3.9 shows this dialog state.

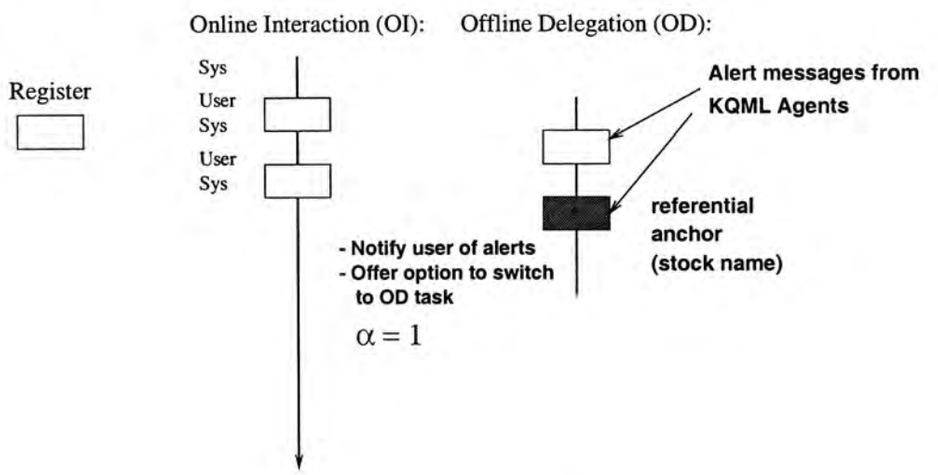

Figure 3.9: Continuing from the Figure 3.8, DM suddenly receives an alert message from the Alert Agent and attaches the incoming alert message into the OD list.

If the user chooses to continue with the current synchronous dialog, DM helps the user to finish the synchronous session. If the user chooses to check

[2]In the example, we use the *status:break* to indicate the previous sub-dialog is completed.

the asynchronous alert message, DM invokes a series of procedures: (i) DM first freezes the current synchronous dialog by pushing the current dialog state into a register, (ii) resets the variable α back to zero, and (iii) DM starts the asynchronous alert message by popping an alert message from the OD list, which is managed in the manner of First-Come-Last-Out(FCLO) in order to ensure that the user gets the most up-to-date alert messages. It should be noted that while handling the alert messages in online interaction, the E-forms belonging to the offline delegation sub-dialogs are marked in order that discourse inheritance does not mix with E-forms belonging to online interaction in the dialog thread. Figure 3.10 illustrates the situation while the user has chosen to handle the asynchronous alert message.

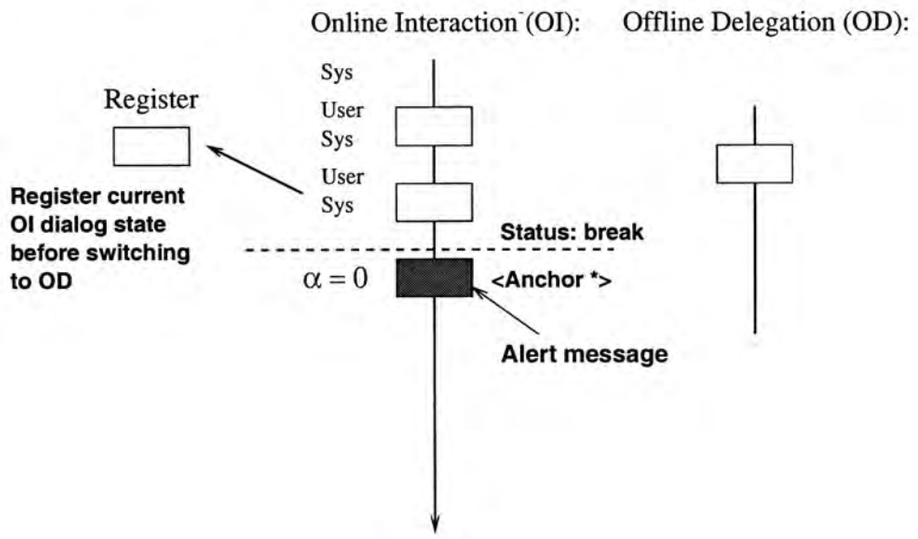

Figure 3.10: Continuing from Figure 3.9, the situation while the user chooses to handle the asynchronous alert message.

After finishing the asynchronous session with several interactive dialog turns, the user can request to revert back to the previous synchronous dialog. If DM detects this query case, it restores the dialog state from register to OI

list. Moreover, DM would invoke a system response reporting a summary of the latest dialog state. Note that DM always reports the number of alert messages in the queue of OD list while the dialog is at the state of completion. Figure 3.11 illustrates the situation where the user requests to restore the previous synchronous dialog state.

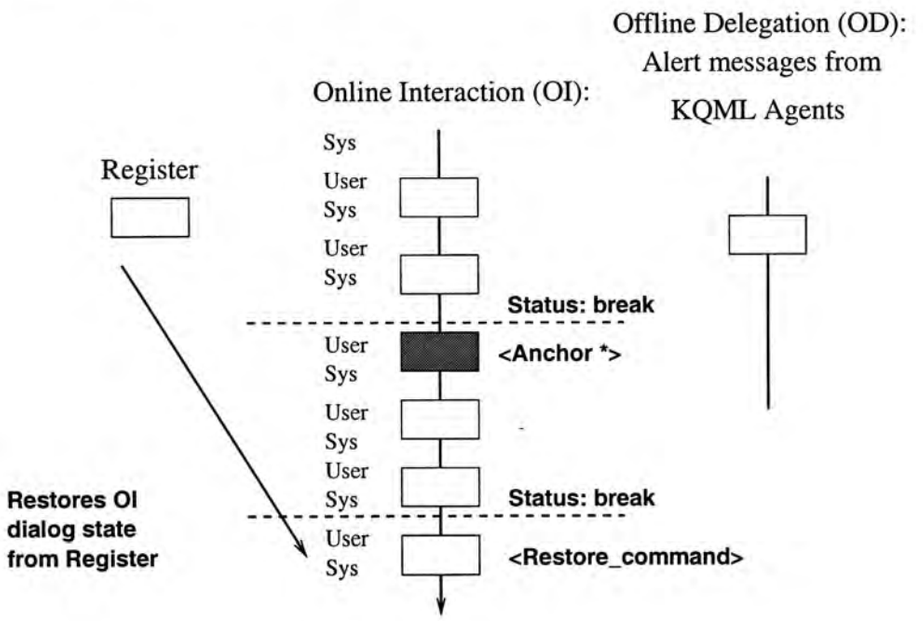

Figure 3.11: Continuing from Figure 3.10, the user requests to restore the previous synchronous dialog.

An example dialog is shown in Table 3.9. After the user says *"Show me the current news of Cheung Kong"*, DM generates the E-form indicating the requested goal is QUOTES and the value of the attribute STOCK_CODE is 0001.HK. As usual, DM begins by appending the E-form to OI list, performs database access and augments the E-form with the retrieved information. Then, DM validates the presence of the retrieved result and updates the E-form with *status:success* to indicate that the user's request has been fulfilled. Response frame generation generates a response frame to send to the

Speech Generation (SG) object, which produces the initial part of the system response *"There is no news for Cheung Kong today."*.

At that same instant the Alert Agent sends a message to DM, regarding a previous buy request for HSBC. DM generates an E-form and invokes a series of steps upon receiving a message from the Alert Agent. DM first appends the message frame to OD list. Since it detects that the previous user request is fulfilled (*status:success*) in the OI list, it invokes the Response frame generation step to produce the system response *"There is one alert message for you regarding a previous buy request on HSBC. If you want to handle the alert message now, please say HSBC. Otherwise, please continue."*. Note that the stock name (HSBC) is used as an anchor word to refer to its corresponding alert message. Moreover, the system sets the value of a global variable $\alpha = 1$ which marks the possible switch from the OI dialog to the OD sub-dialog.

After the user replies with the anchor word "HSBC", DM receives the messages from LU and generates an E-form. DM validates that the α is set (and then resets it); and HSBC is the anchor word for an alert message in OD list. This step also stores the current E-form with *status:to_confirm*, to prepare for a typical transaction's confirmation sub-dialog. A response *"The stock price of HSBC is at ninety-eight dollars presently. Please confirm your previous request from March 20, 2002 at 2:00PM. You wish to buy three lots of HSBC at the requested price of ninety-eight dollars per share. Please say yes to confirm or no to reject the action."* is generated indicating that we have switched to the offline delegation sub-dialog. The user replies 'Yes' and the system responds in the usual procedures.

Then the user requests with the query *"Let's go back"*. LU treats this

as a meta-command to switch back to an online interaction sub-dialog. DM detects this meta-command and restores the latest dialog state (for online interaction) from the register to OI list. Response Generation then presents a summary of this dialog state. As a result the system response *"Previously you requested to see the current news of Cheung Kong but there is no news for Cheung Kong today. How else may I help you?"* is generated.

User1 : *"Show me the news of Cheung Kong."*
LU produces the E-form (semantic frame): GOAL: QUOTES ; STOCK_NAME: Cheung Kong. DM begins by appending the E-form to OI list, performs database access and augments the E-form with the retrieved information (RESULT) to become: GOAL: QUOTES ; STOCK_NAME: Cheung Kong; RESULT: nil. DM then validates the presence of RESULT and updates the E-form with *status:success* to indicate that the user's request has been fulfilled. Response frame generation generates a response frame to send to the Speech Generation (SG) object, which produces the initial part of the system response.
System1 : *"There is no news for Cheung Kong today."*
At this instant the Alert agent sends a message to DM, regarding a previous buy request for HSBC. DM invokes the following series of steps upon receiving a message from the Alert agent. 1. It appends the message frame to OD list. 2. It detects that the previous user request is fulfilled (*status:success*) in the OI E-form. 3. It invokes the response frame generation step to produce the following system response. Notice that the stock name (HSBC) is used as an anchor word to refer to its corresponding alert message. 4. It sets the value of a global variable α=1 which marks the possible switch from the OI dialog to the OD subdialog.
System1 : *"There is one alert message for you regarding a previous buy request on HSBC. If you want to handle the alert message now, please say HSBC. Otherwise, please continue."*

User2 :	*"HSBC."*
LU produces the E-form (semantic frame): GOAL: OOD ; STOCK_NAME: HSBC DM receives this message from LU. Then DM validates that the a is set (and then resets it); and HSBC is the anchor word for an alert message in OD list. This step also stores the current E-form in OI list to a register. It then removes the message frame from OD list and appends to OI list. It also updates the E-form with *status:to_confirm*, to prepare for a typical transaction's confirmation subdialog. Then the system generates the following response (System2). As such we are switching to the OD subdialog.	
System2 :	*"The stock price of HSBC is at ninety-eight dollars presently. Please confirm your previous request from May 20, 2002 at 2:00PM. You wish to buy three lots of HSBC at the requested price of ninety-eight dollars per share. Please say yes to confirm or no to reject the action."*
User3 :	*"Yes."*
LU produces an E-form, and sends it to DM. The system then responds as follows (System3).	
System3 :	*"The buy transaction is completed. Please input a new query."*
User4 :	*"Let's go back."*
LU treats this as a meta-command. DM detects this and restores the latest dialog state (for online interaction) from the register to OI list. Response generation then presents a summary of this dialog state. As such we have switched back to an OI subdialog.	
System4 :	*"Previously you requested to see the past news of Cheung Kong but there is no news for Cheung Kong today. How else may I help you?"*

Table 3.9: An example dialog showing the transitions between the online interaction and offline delegation subdialogs.

3.4 Chapter Summary

This chapter describes our work on mixed-initiative dialog management in ISIS, a trilingual spoken dialog system for the stocks domain. In the dialog

model of ISIS, the system can prompt for missing information, offer price alert services, invoke confirmation sub-dialogs, inherit information from context, reject out-of-domain queries, clarify for doubt information and generate time-out messages to the user. Moreover, the system has a list of meta-commands e.g. *help*, *good-bye*, *undo*, etc., to allow the user to navigate freely in the dialog space. In ISIS, discourse inheritance can inherit values selectively from the discourse for the missing attributes in the current user input. ISIS handles both dialog and discourse by using a form-based approach. Response generation is controlled by a set of templates which state the mapping between the query cases and the corresponding generated response trigger words. Besides, the system can automatically incorporate newly listed stock names (out-of-vocabulary words) into its knowledge base. It can also support online interaction and offline delegation, and the dialog model allows the user to switch freely between the two.

Chapter 4

Understanding Mixed-Initiative Human-Human Dialogs

We have demonstrated the mixed-initiative dialog model of ISIS, in which the shifting of initiatives only depends on the task goal. However, we found that the initiative control should depend on more than just the task goal. Significant efforts have recently been devoted towards understanding mixed-initiative structures in human-human dialogs. Since it is difficult to get human-human dialogs in the ISIS domain, we switched our focus on the restaurants domain. In this chapter, we present our study of the interdependencies among dialog acts, task goals and discourse inheritance in mixed-initiative dialogs in the restaurants domain. In the framework of our study, communication for every request or response is characterized by its categories, dialog act(s) and task goal(s). As the dialog progresses from one turn to the next, selected categories need to be inherited in the discourse and inheritance may be dependent on the task goal or dialog act. The inherited categories augment those in the current (context-dependent) request to help

determine its task goal and dialog act. Finally, we incorporated all of the results of our analysis into a model for discourse inheritance.

4.1 The CU Restaurants Domain

Our study is based on 199 dialogs in the restaurants domain (CU Restaurants), collected from websites and books for English learning [70, 71, 72, 73]. The dialogs capture interactions between the customer and waiter in a restaurant. We divide the corpus into disjoint training and test sets. Table 4.1 presents the distribution of the queries in these two disjoint sets. The average number of waiter and customer dialog turn pairs per one dialog is 5. Table 4.2 shows an example dialog from the corpus. The numbers denote the counter for the dialog turns.

	# customer requests	# waiter responses	# of dialogs
Training set	893	1054	169
Test set	216	266	30
Total	1109	1320	199

Table 4.1: Breakdown of queries and responses in the CU Restaurant domain.

Waiter1	: *"Good evening, sir. How may I help you?"*
Customer2	: *"Can I have the menu, please?"*
Waiter2	: *"Yes, sir. Here. Have you decided on something, sir?"*
Customer3	: *"What is today's special?"*
Waiter3	: *"Abalone soup and stuffed tofu with rice."*
Customer4	: *"I think it would be better to have seafood for a change. I'd like an abalone soup and a grilled fish."*
Waiter4	: *"Anything else, sir?"*
Customer5	: *"No, thanks."*
Waiter5	: *"You're welcome."*

Table 4.2: An example dialog in the CU Restaurants domain.

4.2 Task Goals, Dialog Acts, Categories and Annotation

4.2.1 Task Goals and Dialog Acts

The task goal (TG) shows the domain specifics of the user's request. There are 6 task goals in the CU Restaurants domain as shown in Table 4.3. Definitions of these 6 domain-specific goals are illustrated in Appendix A.

Task Goals	Examples
ASK_INFO	When does your restaurant open?
BILL	Bill, please.
COMPLAINT	This isn't what I ordered!
ORDER	I'd like some chicken curry and rice, a salad and some beer, please.
RESERVATION	Have you a table for two?
SERVING	Could I have some matches, please?

Table 4.3: Six task goals corresponding to the CU Restaurants domain.

The dialog act (DA) expresses the primary communicative intention of the customer's request. We have studied a number of annotation schemes proposed for tagging dialog acts, including VERBMOBIL [74], VERBMOBIL-2 [7], the summer Johns Hopkins LVSCR Workshop-97 summer project (WS97 project) [75, 76, 77], DATE [56, 57, 78], and MITRE [55, 79]. We decided to reference the VERBMOBIL-2 scheme due to the availability of detailed guidelines for tagging and their applicability to our domain.

We have applied the full list of 33 VERBMOBIL-2 dialog acts, as shown in Appendix B for customer requests and waiter responses in our initial tagging. Then, we revised the set of dialog acts, after annotating the entire

training dialog corpus with the full list of VERBMOBIL-2 dialog acts. The following dialog acts were introduced outside of the VERBMOBIL-2 set:

- PREFER
- REQUEST_INFO
- REQUEST_ACTION

We have also renamed some dialog acts from VERBMOBIL-2 or combined some dialog acts together. The following renaming took place under the new scheme or combining with another dialog acts:

- ACCEPT → POSITIVE_FEEDBACK
- DELIBERATE → DEFER
- REJECT → NEGATIVE_FEEDBACK

Moreover, we have omitted some dialog acts from VERBMOBIL-2 since they proved to be subsumable under other dialog acts or simply seldom or never occurred in the training data for both customer and waiter queries. The following dialog acts were removed from the full list:

- CLARIFY
- DEVIATE_SCENARIO
- DIGRESS
- EXPLAINED_REJECT
- FEEDBACK
- GIVE_REASON
- INIT
- NOT_CLASSIFIABLE
- POLITENESS_FORMULA

- REFER_TO_SETTING
- REQUEST
- REQUEST_CLARIFY
- REQUEST_COMMIT

In the restaurants, since the waiter always serves the customer, customer requests and waiter responses may have different sets of dialog acts. For example, the waiter always offers help to the user by saying "May I help you?". However, it is rare for the customer to say "May I help you?" to the waiter. Hence, 14 dialog acts are adopted for customer queries from VERBMOBIL-2. We have shown some dialog acts for the customer responses and the corresponding examples in Table 4.4. Regarding waiter responses, 16 dialog acts are adopted from VERBMOBIL-2. Table 4.5 show some dialog acts for the waiter responses and also the corresponding examples. Detail of the definition and example of each dialog act for customer requests and waiter responses are listed in Appendix C, where some definitions are identical to the ones from VERBMOBIL-2 but others have been modified.

Dialog Acts	Examples
PREFER	I'd like seafood for a change.
REQUEST_ACTION	We'll take the bill now.
REQUEST_INFO	Does that also come with salad?
REQUEST_SUGGEST	What would you like to recommend?

Table 4.4: Dialog acts for customer requests in the CU Restaurants domain.

Dialog Acts	Examples
APOLOGY	We're very sorry for the delay, sir.
COMMIT	We will send your order immediately.
CONFIRM	Your reservation is for a table for you're at 8pm.
OFFER	May i help you?

Table 4.5: Dialog acts for waiter responses in the CU Restaurants domain.

4.2.2 Semantic and Syntactic Categories

We have hand-defined 118 semantic and 3 syntactic categories for punctuations[1], to be extracted from the input customer requests. Examples are shown in Table 4.6. Within this set of categories, 88 are used for inferring the task goal of the customer's request, and 33 are used for inferring the dialog act. The full list of grammar rules for both task goal and dialog act identification is listed in Appendix D.

	Category	→ Terminals
Semantic	Cooking_Style	→ stir fried | marinated | steamed | baked | ···
	Complain	→ complain | complained | complaint | ···
	Thank	→ thanks | thank you | thank you very much | ···
Syntactic	Quest_Mark	→ ?
	Exclam_Mark	→ !
	Period	→ .

Table 4.6: Examples of semantic and syntactic categories in the CU Restaurants domain.

As mentioned earlier, each customer request in our training set is annotated with task goals and dialog acts. We also extracted semantic / syntactic categories from the training query automatically according to the rules such

[1]One may not be able to use punctuation directly if the input request is spoken, but it may be possible to detect related information from the prosody of the utterance.

as those shown in Table 4.6. The training queries are used to train the Belief Networks for automatic identification of the task goals or dialog acts based on the input semantic and syntactic categories.

4.2.3 Annotating the Training Sentences

Based on the definition in VERBMOBIL-2 [7], an *utterance* is an individual unit that corresponds to a dialog act and a task goal. According to the guidelines as stated in Figure 4.1, we segmented the dialog example from Table 4.2 into utterances, as shown in Table 4.7. The numbers in the brackets indicate the corresponding rules for segmentation. For example, the waiter response $Waiter1$ {*"Yes, sir. Here. Have you decided on something, sir?"*} in Table 4.2 can be divided into three utterances, including {*"Yes, sir."*}, {*"Here."*} and {*"Have you decide on something, sir?"*}. In the first utterance, the word *yes* is regarded as a fixed phrase for the dialog act POSITIVE_FEEDBACK. The next utterance *"Here."* with a full stop at the end is classified as a catch-all INFORM utterance, while the last utterance with an independent sentence *"Have you decide one something, sir?"* is classified as a REQUEST_INFO utterance.

In our annotation process, we labeled each utterance with a dialog act and a task goal respectively. In our training corpus, we found cases where a single dialog turn contains multiple utterances. While the task goals of the multiple utterances are always consistent, the dialog acts are not. Hence, each dialog turn is associated with a single task goal, but possibly multiple dialog acts to reflect the user's intention. Additionally, since the waiter always tries to serve the customer in a restaurant, we further assume that in a given

1 An utterance corresponds to a clause or a sentence;
It must contain a finite verb, a verb form that is fully occurs in an independent clause [7].
e.g. "I *want* a seafood platter."

2 For complex sentences with two finite verbs the following rule applies:
a. If one of the verbs is a complement verb which is usually one of the clauses fills either subjects or object position in the complex sentence, taking the other clause as a prepositional argument, then the complex sentence is regarded as one single utterance.
e.g.subject position: "We thought that the play *was* very good."[1]
object position: "I saw the Prime Minister *sleeping.*"[1]
b. Otherwise each of the sub-clauses is regarded as an utterance.

3 There are certain cases in which an utterance does not correspond to a clause as defined in above.
a. Whole turns
Every turn consists of at least one utterance. Therefore, if the material presented as a complete turn does not corresponding to a clause as defined above, it nevertheless is regarded as an utterance.
b. Fixed phrases
Certain dialogue acts can be expressed by more or less fixed lexemes or phrases. These expressions are - if they perform one of the following dialogue acts -regarded as utterances,
e.g.POSITIVE_FEEDBACK: "okay", "yes"
c. Nominal phrases (noun or noun phrases)
A Nominal Phrase can linguistically express the dialogue acts REQUEST_SUGGEST. In such a case the NP is regarded as an utterance.
e.g."chef's suggestion"

[1] http://www.uottawa.ca/academic/arts/writcent/hypergrammar/link.html

Figure 4.1: Utterance definition referenced from VERBMOBIL-2 [7].

Waiter1	:	*"Good evening, sir."* (3b)
Waiter1	:	*"How may I help you?"* (1)
Customer2	:	*"Can I have the menu, please?"* (1)
Waiter2	:	*"Yes, sir."* (3b)
Waiter2	:	*"Here."* (3a)
Waiter2	:	*"Have you decided on something, sir?"* (1)
Customer3	:	*"What is today's special?"* (3c)
Waiter3	:	*"Abalone soup and stuffed tofu with rice."* (3a)
Customer4	:	*"I think it would be better to have seafood for a change."* (2a)
Customer4	:	*"I'd like an abalone soup and a grilled fish."* (1)
Waiter4	:	*"Anything else, sir?"* (3a)
Customer5	:	*"No,"* (3b)
Customer5	:	*"thanks."* (3b)
Waiter5	:	*"You're welcome."* (1)

Table 4.7: The same example dialog shown in Table 4.2 after utterance segmentation. The numbers in the brackets indicate the corresponding rules for segmentation relating Figure 4.1.

dialog turn t, the task goal of the waiter's response is always identical to that of the customer request, i.e. $TG_{Waiter,t} = TG_{Customer,t}$. An example of some annotated customer-waiter interactions from our training set is shown in Table 4.8.

4.3 Selective Inheritance Strategy

4.3.1 Category Inheritance Rules

While annotating our training set, we made the following observations:

1. Given a *context-independent* (self-contained) customer request, the *task goal* can be identified from its semantic and syntactic categories. A *context-dependent* request does not have its full set of categories for de-

Customer1:	*"What is today's special?"* **Categories:** {<What = "what"> <TodaySpecial = "today's special"> <Quest_Mark = "?"> } **Annotated Task Goal:** ORDER **Annotated Dialog Act:** REQUEST_INFO
Waiter1 :	*"Abalone soup and stuffed tofu with rice."* **Categories:** {<Food_Drink = "abalone soup"> <Food_Drink = "stuffed tofu with rice"> <Period = "."> } **Annotated Task Goal:** ORDER **Annotated Dialog Act:** INFORM
Customer2:	*"I think it would be better to have seafood for a change."* **Categories:** {<Preference_Phrase = "I think"> <Food_Drink = "seafood"> <Change = "change"> <Period = "."> } **Annotated Task Goal:** ORDER **Annotated Dialog Act:** NEGATIVE_FEEDBACK
Customer2:	*"I'd like an abalone soup and a grilled fish."* **Categories:** {<Preference_Phrase = "I'd like"> <Food_Drink = "abalone soup"> <Food_Drink = "grilled fish"> <Period = "."> } **Annotated Task Goal:** ORDER **Annotated Dialog Act:** PREFER

Table 4.8: An example dialog segment from Table 4.2 and its task goals, dialog acts and category annotations. Categories are displayed in <*Category* = *terminals*> format.

termining the task goal. For example, the customer query *Customer*2 *"I would try this."* in Table 4.9 is a context-dependent query. With the help of the selective category inheritance, we can identify its task goal from OOD (Out-Of-Domain) to ORDER. We observed that the task goal of the context-dependent query in our training set is always identical to that in the previous dialog turn.

Customer1:	*"What is your recommendation?"* **Task Goal Categories:** {<What = "what"> <Suggest = "recommendation"> } **Task Goal (TG):** ORDER
Waiter1:	*"I'd recommend the filet mignon."* **TG Categories:** {<Suggest = "recommend"> <Food_Drink = "filet mignon"> } **TG:** ORDER
Customer2:	*"That's great!"* **TG Categories:** {NIL } **TG:** OOD
Customer2:	*"I would try this."* **Before Category Inheritance:** **TG Categories:** {<Try = "try"> } **TG:** OOD **After Category Inheritance:** **TG Categories:** {<Try = "try"> *<Food_Drink = "filet mignon">* } **TG:** ORDER

Table 4.9: The customer query (Customer2) "I would try this." is a context-dependent query, where the word "this" refers to the food "filet mignon" in the waiter response Waiter1.

2. The *dialog act* of a customer request can always be identified straight from its categories. We have not seen any context-dependent requests in terms of dialog acts. See Table 4.10 for an illustration.

Customer1:	*"I'd like a seafood platter, please."* **Dialog Act Categories:** { <Preference_Phrase = "I'd like"> <Please = "please"> <Period = "."> } **Dialog Act (DA):** INFORM (This query is self-contained and states the user's preference.)
Waiter1 :	*"Anything else, sir?"* **DA Categories:** {<Quest_Mark = "?"> } **DA:** REQUEST_INFO
Customer2:	*"What would you recommend?"* **DA Categories:** {<Wh_Word = "what"> <Suggest = "recommend"> <Quest_Mark = "?"> } **DA:** REQUEST_SUGGEST (This query is self-contained and asks for suggestions.)

Table 4.10: An example dialog illustrating that category inheritance is not required for dialog act identification.

3. *Category inheritance* in the CU Restaurants corpus involves inheriting appropriate categories from the previous dialog turn(s) to the current dialog turn. In particular, the dialogs in our corpus seem to indicate that it is sufficient to consider only the *previous* dialog turn and its inherited discourse. Furthermore, categories that need to be inherited largely depend on the task goal. By considering the task goal of the current customer request, we can determine the appropriate categories to inherit.

Based on these observations, we wrote 5 *selective* inheritance rules. Each rule corresponds to one of the task goals in the CU Restaurants domain, except for the goal of ASK_INFO, which requires no inheritance (see Table 4.11 for an example).

Customer1:	*"What kind of food do you serve?"* **TG Categories:** {<What = "what"> <Food = "food"> <Serve = "serve"> } **TG:** ASK_INFO
Waiter1 :	*"We serve a great variety of popular Japanese dishes."* **TG Categories:** {<Serve = "serve"> <RelativeAmount = "variety"> <Country = "Japanese"> <Dish = "dishes"> } **TG:** ASK_INFO
Customer2:	*"When does the restaurant open for breakfast?"* **TG Categories:** {<When = "when"> <Restaurant = "restaurant"> <Open = "open"> <MealDescription = "breakfast"> } **TG:** ASK_INFO
Waiter2 :	*"7 a.m., sir."* **TG Categories:** {<Time = "7 a.m."> } **TG:** ASK_INFO

Table 4.11: An example dialog showing the queries with task goal ASK_INFO requires no category inheritance.

If $TG_{Customer,t}$ = BILL,

capture values for categories <Bill>, <HowMuch> and <PriceValue> in the current query.

If current values are absent,

inherit from $Waiter_{t-1}$ or $Customer_{t-1}$

($Waiter_{t-1}$ has higher priority since it follows $Customer_{t-1}$)

Figure 4.2: Category inheritance rule for the task goal BILL.

Figure 4.2 shows the format of the category inheritance rules. The category inheritance rules for the task goals BILL, COMPLAINT, ORDER, RESERVATION and SERVING are described below:

- Inherit the categories <Bill>, <HowMuch> and <PriceValue> for the BILL queries (i.e., an inquiry about billing).
- Inherit the categories <Complain>, <Criticism>, <Course> and <MealDescription> for COMPLAINT queries (i.e., an inquiry about handling the complaints).
- Inherit the category <MealDescription> from the previous waiter or customer dialog turns and inherit the category <Food_Drink> only if the current query does not contain the category <TodaySpecial> for ORDER queries (i.e., an inquiry about food ordering).
- Inherit the categories <Arrange>, <Location>, <MealDescription>, <NumberValue>, <RelativeDate>, <RelativeTime>, <Reserve>, <SmokeOption>, <Table> and <Time> for the customer queries with task goal RESERVATION (i.e., an inquiry about table reservation).
- Inherit the category <Utensil> from previous waiter dialog turn if the current customer query contains the category <NumberValue> and inherit the category <Food_Drink> if the current query contains the category <Cook> for SERVING queries (i.e. an inquiry about reqesting utensils or menu).

As an example, the category inheritance rule for the BILL queries is illustrated in Table 4.12 with a dialog example. In the example, the categories <Bill>, <HowMuch> and <PriceValue> are selectively inherited in

the query "Here it is", indicating the customer is paying a bill with the price one hundred fifty dollars. The examples for the category inheritance rules of the task goal COMPLAINT, ORDER, RESERVATION and SERVING are described in Appendix E.

Customer1:	*"Let me have the bill, please."* **TG Categories:** {<Bill = "bill">} **TG:** BILL
Customer1:	*"How much is it?"* **TG Categories:** {<HowMuch = "how much">} **TG:** BILL
Waiter1 :	*"Thank you, sir."* **TG Categories:** {NIL } **TG:** OOD
Waiter1 :	*"One hundred fifty dollars, please."* **TG Categories:** { <PriceValue = "one hundred fifty dollars"> } **TG:** BILL
Customer2:	*"Here it is."* **TG Categories:** {<Here = "here"> *<Bill = "bill"> <HowMuch = "how much">* *<PriceValue = "one hundred fifty dollars">* } **TG:** BILL

Table 4.12: Categories <Bill>, <HowMuch> and <PriceValue> are selectively inherited for queries with task goal BILL. The categories in italics are inherited from discourse.

4.3.2 Category Refresh Rules

While the selective inheritance rules are applied, *over-inheritance* is found in some training dialog turns. Over-inheritance occurs in the dialogs with confirmation (POSITIVE_FEEDBACK) or rejection (NEGATIVE_FEEDBACK). Table 4.13 shows an example dialog where over-inheritance of the category

<Food_Drink> occurred after the customer rejected (NEGATIVE_FEEDBACK) the waiter's suggestion. This over-inheritance causes the system to misunderstand the customer to say that he also prefers the suggested "mushrooms".

Waiter1 :	*"Today we have fresh mushrooms too."* **Categories:** { <RelativeDate = "today"> <Food_Drink = "mushrooms"> <Period = "."> } **TG:** ORDER (from annotation) **DA:** SUGGEST (from annotation)
Customer2:	*"I prefer something else."* **Categories:** { <Preference_Phrase = "prefer"> <Else = "something else"> <Period = "."> ***<Food_Drink = "mushrooms"> (Over-inheritance)*** } **TG:** ORDER **DA:** NEGATIVE_FEEDBACK

Table 4.13: An example dialog showing over-inheritance of the category <Food_Drink> for queries with task goal ORDER. This over-inheritance causes the system to misunderstand that the customer also prefers the suggested "mushrooms".

If $TG_{Customer,t}$ = ORDER AND
$DA_{Waiter,t-1}$ = SUGGEST AND
($DA_{Customer,t}$!= POSITIVE_FEEDBACK OR $DA_{Customer,t}$ = NEGATIVE_FEEDBACK),
disinherit the concept <Food_Item> from $Waiter_{t-1}$

Figure 4.3: Refresh rule for the goal ORDER.

The example in Table 4.13 indicates that although the category inheritance is largely dependent on the task goal of the customer query, inheritance may also be influenced by the dialog act; i.e. both the task goal and dialog act together influence the category inheritance. Hence, we have developed 4

$TG_{Customer,t}$	$DA_{Customer,t}$	$DA_{Waiter,t-1}$	Disinherited Categories
ORDER	NEGATIVE_FEEDBACK	SUGGEST	<Food_Drink>
RESERVATION	NEGATIVE_FEEDBACK	SUGGEST	<Location>
RESERATION	NEGATIVE_FEEDBACK	SUGGEST	<SmokeOption>
$TG_{Customer,t}$	$DA_{Customer,t-1}$	$DA_{Waiter,t}$	Disinherited Categories
BILL	REQUEST_INFO	NEGATIVE_FEEDBACK	<PriceValue>

Table 4.14: Four refresh rules that specify categories to disinherit given the task goals and dialog acts. As an example, the first rule specifies disinheriting the category <Food_Drink> for an ORDER query if the customer previously rejected ($DA_{Customer,t}$ = NEGATIVE_FEEDBACK) the suggestion from the waiter ($DA_{Waiter,t-1}$ = SUGGEST).

refresh rules to undo over-inheritance. These rules incorporate the interdependencies among task goals, dialog acts, semantic and syntactic categories. Figure 4.3 shows an example format of the category refresh rules. Table 4.14 summarizes the four refresh rules. The category refresh rules are described below:

- Disinherit the category <Food_Drink> for ORDER queries if the customer has previously rejected (i.e. the dialog act of the current query is identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previous waiter response is SUGGEST), as in the example dialog shown in Table F.1.

- Disinherit the category <Location> for RESERVATION queries if the customer has previously rejected (i.e. the dialog act of the current query is identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previous waiter response is SUGGEST).

- Disinherit the category <SmokeOption> for RESERVATION queries if the customer has previously rejected (i.e. the dialog act of the current query is identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previous waiter response is SUGGEST).

- Disinherit the category <PriceValue> for BILL queries if the customer clarifies the bill value (i.e. the dialog act of the previous query is identical to REQUEST_INFO) but the waiter has not explicitly agreed with the proposed bill value (i.e. the dialog act of waiter response is NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK).

The category refresh rules for the category <Location> is illustrated in Table 4.15 with an example dialog . In the example, the customer first requests a table for two in the *main restaurant.* However, the tables are filled up. Hence, the waiter suggested the customer whether he prefers to have a table at the *coffee shop* instead. The customer rejected the suggestion and replied he would like to wait instead. In the example, the category <Location> is over-inherited in the customer replied query Customer2. This causes the system to misunderstand that the customer also prefers the suggested location "coffee shop". In Table F.2 in Appendix F, the categories in italics are inherited from discourse. The examples for the other refresh rules are described in Appendix F.

Customer1:	*"I'd like to have a table for two in the main restaurant."* **Categories:** { <Preference_Phrase = "I'd like"> <Table = "table"> <NumberValue = "two"> <Period = "."> <Location = "main restaurant"> } **TG:** RESERVATION **DA:** PREFER
Waiter1 :	*"I'm afraid all our tables are filled up right now, sir. If you are in a hurry, we also serve dinner at coffee shop."* **Categories:** { <Table = "tables"> <Full = "filled up"> <RelativeTime = "right now"> <Period = "."> <Serve = "serve"> <MealDescription = "dinner"> <Location = "coffee shop"> <Period = "."> } **TG:** RESERVATION (from annotation) **DA:** INFORM, SUGGEST (from annotation)
Customer2:	*"No, thanks. I'll wait then."* **Categories:** { <No_Phrase = "no"> <Thank_Phrase = "thanks"> <Period = "."> <Wait = "wait"> <Period = "."> *<Table = "table">* *<NumberValue = "two"> <RelativeTime = "right now">* *<MealDescription = "dinner">* ***<Location = "coffee shop"> (Over-inheritance)***} **TG:** RESERVATION **DA:** NEGATIVE_FEEDBACK, THANK, INFORM

Table 4.15: Over-inheritance of the category <Location> for queries with task goal RESERVATION.

4.4 Task Goal and Dialog Act Identification

4.4.1 Belief Networks Development

We used Belief Networks (BNs) to infer the task goal and dialog act(s) for a customer request based on its inherent and inherited categories. The detailed experimental setup is adapted from that used in our previous work for the ATIS domain [54]. We have trained 6 BNs for each task goal. Each BN is used to infer the presence / absence of its corresponding task goal, based on the input categories. The binary decisions across all BNs are combined to identify the task goal of the customer request. If all BNs vote negative for their respective goals, the request may be context-dependent or out-of-domain (OOD). Similarly, we have trained 13 BNs to identify the dialog act for each customer request except for INFORM. If all 13 BNs vote negative, the dialog act is set to INFORM - a catch-all category as used in VERBMOBIL-2.

BN is a probablistic causal network. We trained our BNs with the entire training corpus. Table 4.16 and Table 4.17 show the distribution of utterances with different task goals and dialog acts in the training set. In our implementation, we have used a simple topology of BNs which is identical to a naïve Bayes setup. Figure 4.4 illustrated a simple BN for the task goal ORDER. In this Bayesian topology, the categories are assumed to be independent of one another, but are dependent only on the goal. Directed arrows are drawn from cause to effect.

Task Goal	Number of utterances in training set
ASK_INFO	19
BILL	71
COMPLAINT	34
ORDER	394
RESERVATION	96
SERVING	15

Table 4.16: Distribution of utterances with different task goals in the training set.

Dialog Act	Number of utterances in Training set
BACKCHANNEL	67
BYE	23
DEFER	25
NEGATIVE_FEEDBACK	67
POSITIVE_FEEDBACK	225
GREET	29
PREFER	297
REQUEST_ACTION	39
REQUEST_COMMENT	32
REQUEST_INFO	111
REQUEST_SUGGEST	25
SUGGEST	29
THANK	100

Table 4.17: Distribution of utterances with different dialog acts in the training set.

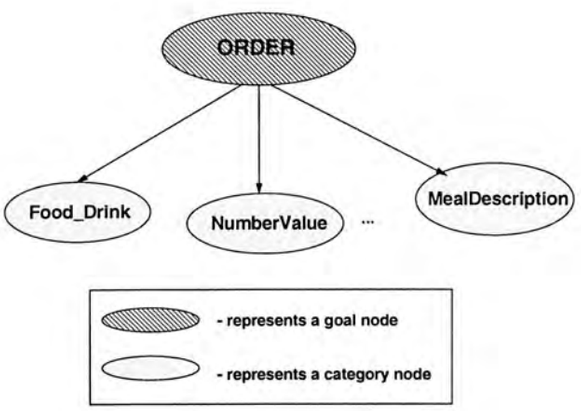

Figure 4.4: A simplified BN for the task goal ORDER.

4.4.2 Varying the Input Dimensionality

In our work, a series of experiments were conducted in which we varied the BN input dimensionality, which is equivalent to the number of stored categories per goal. Variation should cover the range from 10 categories to the full set of 88 categories for task goal identification and from 10 categories to the full set of 33 categories for dialog act identification. The task goal and dialog act identification accuracies for the training set are tabulated in Figure 4.5 and Figure 4.6 respectively.

Performance accuracies in the plot are based on the average goal identification accuracies for the training set. As observed in both figures, the training accuracies generally tend to increase with input dimensionality. In Figure 4.5, task goal identification accuracies of the training set are the highest at 15 categories per goal but it tends to decrease and stablizes beyond 20 categories, possibly due to the overfitting of the training data. On the other hand, dialog act identification accuracies for the training set tend to increase and stabilize beyond 20 categories per goal as illustrated in Figure 4.6. Since the stablization suggests the suitable parameter setting for goal identification, we select the BNs with 15 input nodes for task goal identification and the BNs with 20 input nodes for dialog act identification.

4.4.3 Evaluation

In the experiments, we used the text-based customer queries as input to the system on the basis of an assumed perfect recognition. Evaluation indicated that 97.8% of the test set *utterances* have correctly identified dialog acts. Since task goal identification can be affected by category inheritance, we

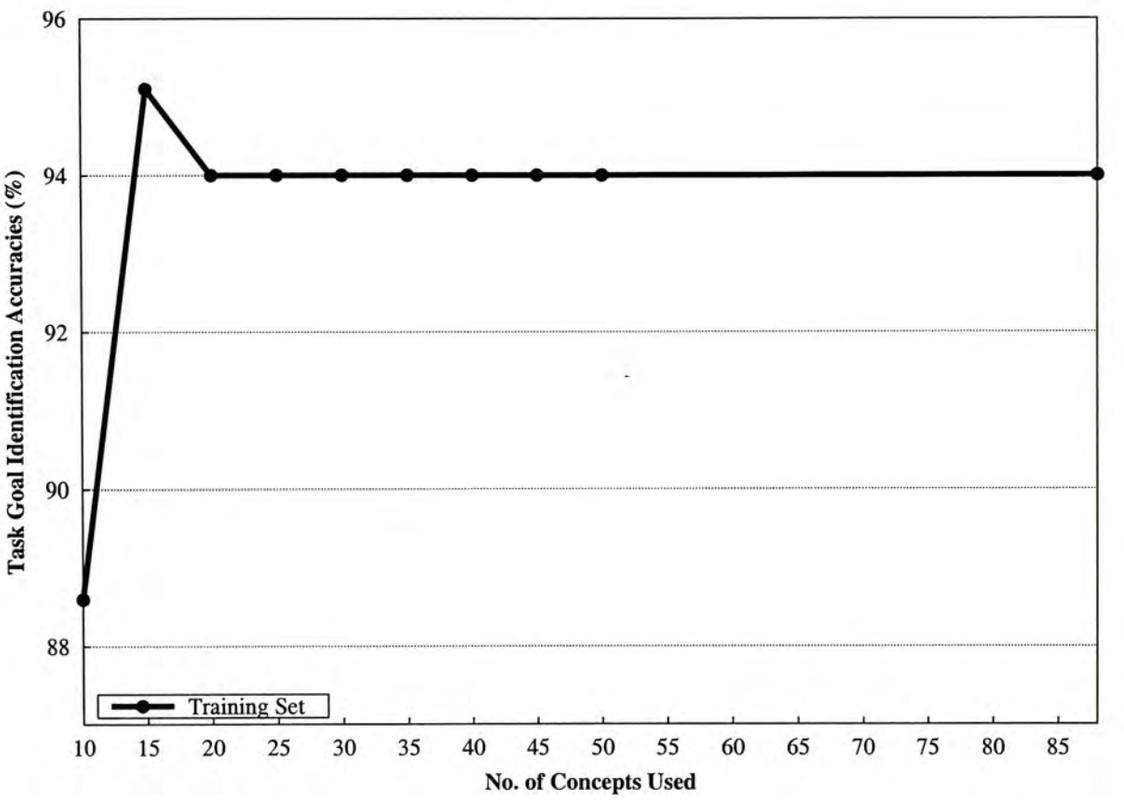

Figure 4.5: Task goal identification accuracies for different BN input dimensionalities schemes.

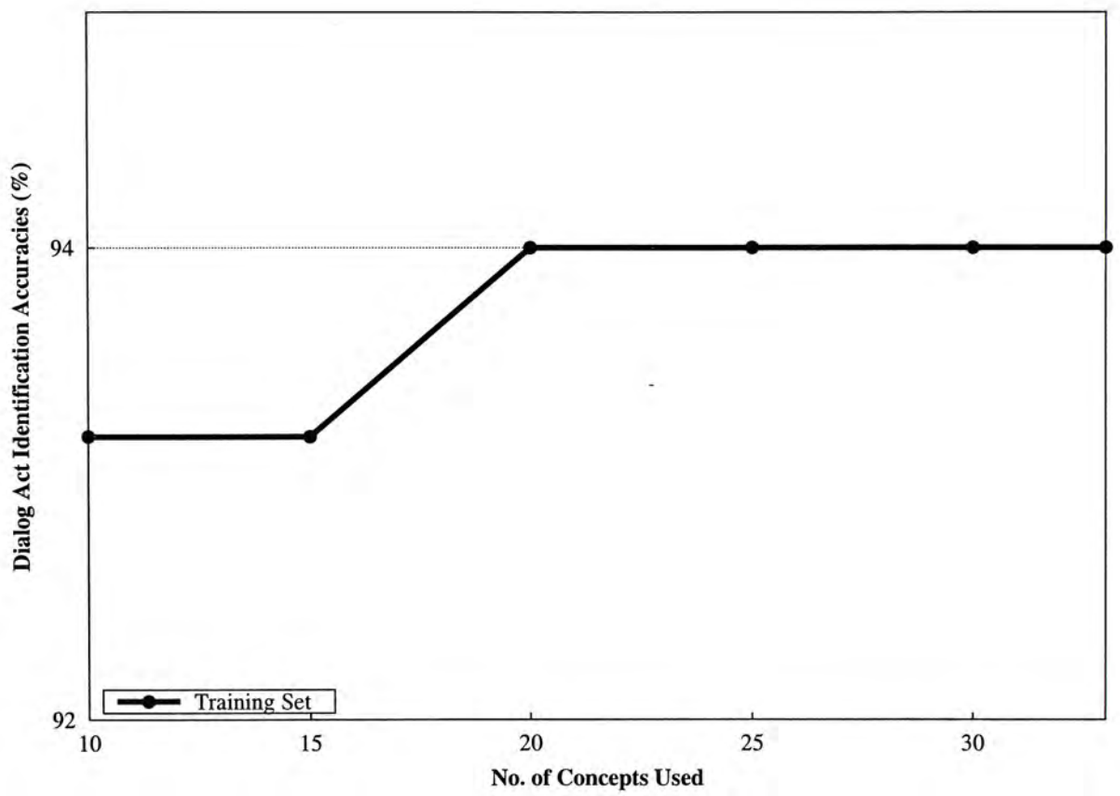

Figure 4.6: Dialog act identification accuracies for different BN input dimensionalities schemes.

experimented with three inheritance strategies, including the selective strategy and two other control strategies - (i) no categories are inherited; and (ii) all categories are inherited throughout the dialog session. We trained a set of BNs for each inheritance strategy. We also applied the corresponding inheritance strategy to the test set. Figure 4.7 shows the average task goal identification performance for the three inheritance strategies based on the test set. Performance in task goal identification is measured in terms of the *dialog turns*, because each turn has a single task goal only. Selective inheritance gave the best performance. Table 4.18 provides an example of how the selective inheritance strategy gave rise to the correct task goal, but other strategies identified an incorrect task goal. The categories in italics are inherited from discourse.

In error analysis, we find that failures in both the task goal and dialog act identification are due to insufficiencies in our category tagging. For example, some words are tagged to incorrect categories, or are missing from our grammar rules derived from the training set. Moreover, the BNs often label context-dependent customer requests wrongly as OOD and led to failures in task goal identification.

4.5 Procedure for Discourse Inheritance

We have developed the following discourse inheritance procedure for handling customer requests in our mixed-initiative dialog corpus. Discourse inheritance includes both category inheritance and task goal inheritance. Table 4.19 describes the discourse inheritance procedure.

Step 2 specifies *task goal inheritance*, which we found to be necessary for

Waiter1:	*"Your seafood platter costs one hundred dollars. I'm afraid the voucher cannot cover the cost of the meal. Would you mind paying the extra in cash please?"* **TG Categories:** { <Food_Item = "seafood platter"> <Cost = "costs"> <PriceValue = "one hundred dollars"> <PayMethod = "voucher"> <Cost = "cost"> <MealDescription = "meal"> <Pay = "paying"> <Extra = "extra"> <PayMethod = "cash"> } **Annotated TG:** BILL

Customer2:	*"OK. But how much is the voucher worth?"*
	
	TG Categories from <u>No Inheritance</u>: {
	<HowMuch = "how much"> <PayMethod = "voucher"> }
	Inferred TG: OOD (out-of-domain)
	
	TG Categories from <u>Selective Inheritance</u>: {
	<HowMuch = "how much"> <PayMethod = "voucher">
	<PriceValue = "one hundred dollar"> }
	Inferred TG: BILL
	
	TG Categories from <u>All Inheritance</u>: {
	<HowMuch = "how much"> <PayMethod = "voucher">
	<Food_Item = "seafood platter"> <Cost = "costs">
	<PriceValue = "one hundred dollars">
	<PayMethod = "voucher">
	<Cost = "cost"> <MealDescription = "meal">
	<Pay = "paying"> <Extra = "extra">
	<PayMethod = "cash"> }
	Inferred TG: OOD

Table 4.18: Different task goal identification results of the same customer request with different category inheritance strategies. In the strategy of no inheritance, the categories of the query Customer2 are not strong enough to infer the correct task goal BILL. In the strategy of all inheritance, the BNs are confused by so many categories inherited from the discourse. In the example, only the one with selective inheritance strategy got the correct task goal identification. The categories in italics are inherited from discourse.

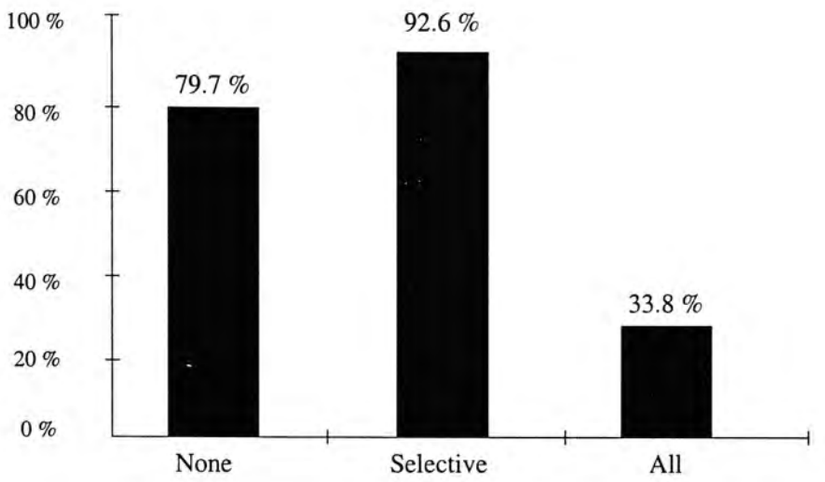

Figure 4.7: Test set accuracies for task goal identification based on different category inheritance strategies.

Step 1	Parse for **categories** in the incoming customer request ($C_{Customer,t}$)
Step 2	Infer the **task goal** ($TG_{Customer,t}$) of the request using the trained BNs and $C_{Customer,t}$ If $TG_{Customer,t}$ = nil (all BNs vote negative), then $TG_{Customer,t} = TG_{Customer,t-1}$ (treat request as context-dependent, perform **task goal inheritance** based on previous dialog turn).
Step 3	Infer the **dialog act** ($DA_{Customer,t}$) of the request using the trained BNs and $C_{Customer,t}$ If $DA_{Customer,t}$ = nil (all BNs vote negative), $DA_{Customer,t}$ = INFORM
Step 4	Apply selective **category inheritance rules** based on $TG_{Customer,t}$ and $DA_{Customer,t}$. and **category refresh rules** based on $TG_{Customer,t}$, $DA_{Customer,t}$, $DA_{Customer,t-1}$, $DA_{Waiter,t}$ and $DA_{Waiter,t-1}$.

Table 4.19: Discourse inheritance procedure determined based on the CU Restaurants Corpus.

context-dependent customer requests. The BNs often label these requests as OOD, i.e. all BNs vote negative for their corresponding task goals. Under such circumstances, we inherit the task goal of the previous dialog turn, i.e. $TG_{Customer,t} = TG_{Waiter,t-1}$. With task goal inheritance, we improved task goal identification of the test set from 92.6% (see Figure 4.7) to 93.2%.

The *discourse inheritance procedure* incorporates our findings in the interdependencies among task goals, dialog acts and category inheritance. Evaluation shows that this procedure correctly handled 95.9% of *dialog turns* in our test set.

4.6 Chapter Summary

This chapter describes a study of interdependencies among dialog acts, task goals and discourse inheritance in mixed-initiative dialogs in the CU Restaurants domain. Our study is based on 199 dialogs in the restaurants domain, with disjoint training (169 dialogs) and test sets (30 dialogs). Our training set is first annotated manually in terms of the task goals and dialog acts and tagged automatically in terms of the semantic and syntactic categories for each customer request and waiter response. Based on the annotation process, we observed the following:

- The task goal of a context-independent customer request can be identified from its categories, while the task goal of a context-dependent request can be inherited from the previous dialog turn.
- Dialog act identification does not require category inheritance, while task goal identification does.

- Category inheritance is largely dependent on the task goals of the current query. However in some cases it is also influenced by the dialog act in the cases of confirmation or rejection.

We have written a set of category inheritance and refresh rules, which constitute our *selective inheritance strategy.* We used Belief Networks (BNs) to automate identification of task goals and dialog acts based on input categories. Selective category inheritance outperformed two alternate control strategies where none or all of the categories are inherited. The selective strategy achieved correct task goal identification for 92.6% of the dialog turns and correct dialog act identification for 97.8% of the utterances in the test set. We have also developed a *discourse inheritance procedure*, which can correctly handle 95.9% of the dialog turns in the test set.

Chapter 5

Cooperative Response Generation in Mixed-Initiative Dialog Modeling

Aside from the automatic analysis of the user's requests, we also explore how to respond in a mixed-initiative fashion. The categories, task goal(s) and dialog act(s) from the request should be useful for the automatic generation of a coherent response. Hence, in this chapter, we present our model for cooperative response generation. We leverage from our analysis results in the CU Restaurants domain to a rule-based dialog system, which generates cooperative response based on the identified task goals, dialog acts and categories.

5.1 System Overview

Figure 5.1 depicts the overview of our dialog system in the CU Restaurants domain. In our computational model, we first implemented the *discourse inheritance procedure.* Hence at this point, the information of task goal ($TG_{Customer,t}$), dialog act ($DA_{Customer,t}$) and categories ($TGC_{Customer,t}$ and $DAC_{Customer,t}$) of the incoming customer request is available for our response generation process. In the following, we focus on how to generate a cooperative system response to the customer request. Our system first generates a state space representation from the *state space generation* process. Then based on the *matching rules* generated from our training corpus, we get the corresponding task goal $TG_{Waiter,t}$ and dialog act $DA_{Waiter,t}$ of the system response. Through the *response frame generation* process, we generate the corresponding system response frame based on the *hand-defined* rules. Finally from the *text generation* process, the text response is generated from the response frame. Hence, the customer receives the corresponding waiter response via the text interface. Additionally, we save all the information of the user request $Customer_t$ and the generated system response $Waiter_t$ in the *discourse* in each turn. This procedure repeats for the next incoming customer request. In the following, we will present each part of our system in detail.

5.1.1 State Space Generation

The *state space generation* process is responsible for generating a state space representation for the customer requests in each dialog turn. The *state space representation* represents knowledge of the current state, which should pro-

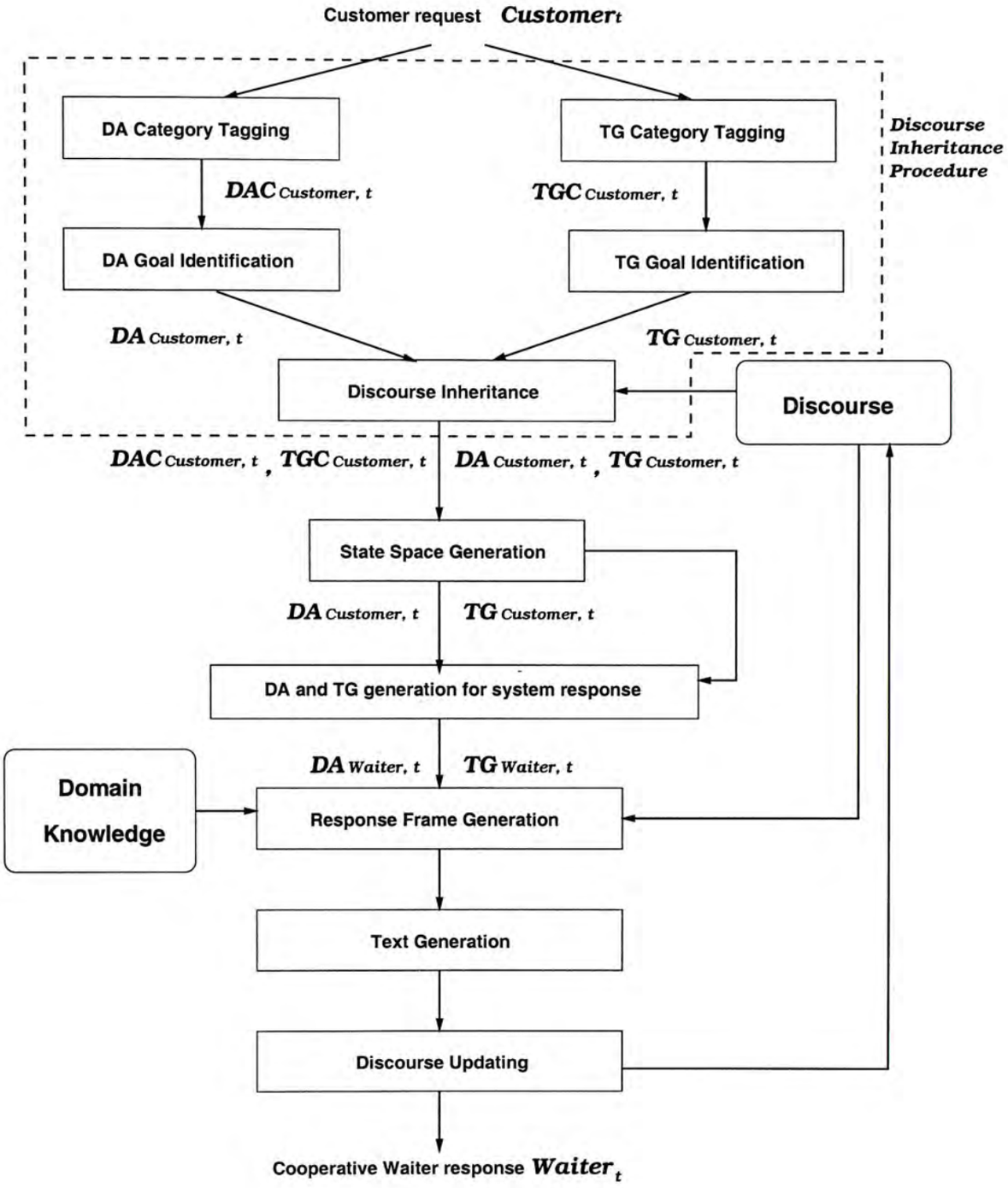

Figure 5.1: Computational framework for generating cooperative responses in the CU Restaurants domain. In this model, we have implemented the discourse inheritance procedure resulting from the study in Chapter 4. Here, DA indicates the dialog act while TG indicates the task goal. Also, DAC and TGC indicate the categories tagged for dialog act identification and task goal identification respectively.

vide sufficient information for the current request. Additionally, the size of state space should be kept small enough for the problem to remain tractable. In our approach, our state space representation is constructed by *two* variables, including the inferred task goal $TG_{Customer,t}$ and the inferred dialog act $DA_{Customer,t}$ of the current customer query. Table 5.1 summarizes the possible values used in the state space representation. In our dialog corpus, each customer dialog turn may contain multiple utterances, and hence multiple task goals and dialog acts. In this case we only consider the *latest* task goal and dialog act in each customer dialog turn for response generation. Table 5.2 shows an example customer request and the corresponding state space representation.

Variables	Possible values
$TG_{Customer,t}$	ASK_INFO, BILL, COMPLAINT, ORDER, RESERVATION, SERVING, OOD
$DA_{Customer,t}$	BACKCHANNEL, CLOSE, DEFER, GREET, INFORM, NEGATIVE_FEEDBACK, POSITIVE_FEEDBACK, REQUEST_ACTION, REQUEST_COMMENT, REQUEST_INFO, REQUEST_SUGGEST, PREFER, SUGGEST, THANK

Table 5.1: Possible values used in state space representation.

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}

Table 5.2: An example customer request and the corresponding state space representation.

5.1.2 Task Goal and Dialog Act Generation for System Response

With the state space representation as an input, the system generates the corresponding task goal and dialog act of the waiter response accordingly. Based on our 169 training dialogs, we have generated a rule for every dialog turn which states the dialog transition from the customer request to the system response. There are 853 waiter and customer dialog turn pair. Hence in total, we have generated 853 rules as shown in Figure 5.2. However, we found that a waiter dialog turn, similar to a customer turn, may contain multiple utterances. While the task goals of the multiple utterances are always consistent, the dialog acts are not. Hence, each waiter dialog turn is associated with a single task goal, but multiple dialog acts jointly reflect the waiter's intention. Moreover, even if the customer presents the same request, the waiter may respond to the customer in different ways in different dialogs as seen from the our training corpus. Table 5.3 shows cases where the waiter responds differently for the same customer request "*Let me have the bill, please.*". In this example, these waiter responses result in different rules but the same state space from the same customer request. In order to standardize our output, for each input state space we selected one dialog act of the waiter response. Figure 5.2 shows 10 rules generated by 10 dialog turns with the same state space from the customer requests and the same task goal and dialog act generated from the waiter response. In order to select one dialog act for the state space {BILL, REQUEST_ACTION}, we condense the number of *priority* and *occurrence* of the corresponding dialog act in each rule. Figure 5.1.2 shows the examples of the condensed rules. In the rule BILL , REQUEST_ACTION -> BILL , THANK&INFORM stated

in Figure 5.2, since the dialog act THANK is the first dialog act of the waiter response in this dialog turn pair, the dialog act THANK is placed in the first priority. Similarly, the dialog act INFORM is placed in the second priority in the waiter response. However, the number of occurrences of both cases are ten. We condensed the rules based on the priority and the number of occurrences. As shown in Figure 5.3, these two generated dialog acts are separated with different rules with different priority. Finally, we incorporated all the condensed rules with the highest priority and the greatest number of occurrence into the final set. There are 96 rules in our final set. Some rules are shown in Figure 5.4. Table 5.5 continues the example in Table 5.2 and shows the generated system task goal and dialog act.

5.1.3 Response Frame Generation

Response frame generation generates response frames for the waiter response. Each response frame is mapped with one response trigger word which indicates the corresponding waiter response type. Based on the waiter responses in the training corpus, we first hand-defined 351 rules for generating response trigger words. In these 351 rules, 216 rules are used for general response generation while the remaining 135 rules are used for cooperative response generation. According to the rules for the allocation of control referenced from Whittaker et al [8, 9] (see Table 2.1 in Chapter 2), 69 rules cause a shift of initiative control from the user to the system. As an example, rule (1) in Figure 5.5 indicates that if the generated task goal ($TG_{Waiter,t}$) and dialog act ($DA_{Waiter,t}$) of the waiter response are BILL and THANK respectively, then the response trigger word for general response generation (FIRST_RESPONSE_TYPE) should be "thank". With the same generated task goal and dialog act of the waiter response, if the dialog act ($DA_{Waiter,t}$)of customer request is REQUEST_ACTION, the system should cooperatively present the value of the bill. Hence the response trigger word for cooperative response genera-

$TG_{Customer,t}$, $DA_{Customer,t}$ -> $TG_{Waiter,t}$, $DA_{Waiter,t}$
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , THANK&INFORM
BILL , REQUEST_ACTION -> BILL , INFORM
BILL , REQUEST_ACTION -> BILL , INFORM
BILL , REQUEST_ACTION -> BILL , INFORM
BILL , REQUEST_ACTION -> BILL , REQUEST_INFO
BILL , REQUEST_ACTION -> BILL , REQUEST_INFO
BILL , REQUEST_ACTION -> BILL , REQUEST_INFO
BILL , REQUEST_ACTION -> BILL , REQUEST_INFO ...

Figure 5.2: Rules generated from 10 dialog state transitions with the same state space but different task goal and dialog act for waiter response. In the example, $TG_{Customer,t}$ and $DA_{Customer,t}$ indicate the task goal and dialog act of the customer request whereas $TG_{Waiter,t}$ and $DA_{Waiter,t}$ indicate those of the waiter response.

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}
Waiter1	Case 1: Thank you, sir. Your bill goes to two hundred dollars. **Task Goal (TG):** BILL **Dialog Act (DA):** THANK, INFORM **Rule:** BILL, REQUEST_ACTION → BILL, THANK&INFORM Case 2: Here it is. Two hundred dollars, sir. **Task Goal (TG):** BILL **Dialog Act (DA):** INFORM, INFORM **Rule:** BILL, REQUEST_ACTION → BILL, INFORM&INFORM Case 3: Certainly, sir. Two hundred dollars, please. **Task Goal (TG):** BILL **Dialog Act (DA):** POSITIVE_FEEDBACK, INFORM **Rule1:** BILL, REQUEST_ACTION → BILL, POSITIVE_FEEDBACK&INFORM Case 4: Your bill goes to two hundred dollars. **Task Goal (TG):** BILL **Dialog Act (DA):** INFORM **Rule:** BILL, REQUEST_ACTION → BILL, INFORM Case 5: Would you like your check? **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_INFO **Rule:** BILL, REQUEST_ACTION → BILL, REQUEST_INFO

Table 5.3: Different waiter responses with the same customer request. The number of rules indicate the priority of the dialog act generated in the corresponding waiter response.

$TG_{Customer,t}$, $DA_{Customer,t}$ -> $TG_{Waiter,t}$, $DA_{Waiter,t}$ — priority_occurrence
BILL , REQUEST_ACTION -> BILL , THANK — 1_10
BILL , REQUEST_ACTION -> BILL , INFORM — 2_10 ...

Figure 5.3: Examples of the condensed rules, where the priority and occurrence of each generated dialog act are indicated in "priority_occurrence" format.

$TG_{Customer,t}$, $DA_{Customer,t}$ -> $TG_{Waiter,t}$, $DA_{Waiter,t}$ — priority_count
BILL , BACKCHANNEL -> BILL , OFFER — 1_1
BILL , CLOSE -> BILL , CLOSE — 1_11
BILL , DEFER -> BILL , OFFER — 1_1
BILL , FEEDBACK_POSITIVE -> BILL , INFORM — 1_1
BILL , FEEDBACK_NEGATIVE -> BILL , INFORM — 1_1
BILL , GREET -> BILL , GREET — 1_1
BILL , INFORM -> BILL , THANK — 1_7
BILL , PREFER -> BILL , INFORM — 1_1
BILL , REQUEST_ACTION -> BILL , THANK — 1_10
BILL , REQUEST_COMMENT -> BILL , FEEDBACK_POSITIVE — 1_1
BILL , REQUEST_INFO -> BILL , INFORM — 1_7
BILL , REQUEST_SUGGEST -> BILL , INFORM — 1_11
BILL , SUGGEST -> BILL , THANK — 1_1
BILL , THANK -> BILL , THANK — 1_11 ...

Figure 5.4: Example rules for generating task goal and dialog act of the system response in our final set.

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}
Waiter1	**Generated TG of the system:** BILL **Generated DA of the system:** THANK

Table 5.4: The system generates the task goal BILL and the dialog act THANK for the waiter response. This table continues the example in Table 5.2.

tion (NEXT_RESPONSE_TYPE) should be "bil_inform_bill" which is indicated by the rule (2) in Figure 5.5. Since the priority of the tag FIRST_RESPONSE_TYPE is higher than that of NEXT_RESPONSE_TYPE, we generated the response trigger word "thank" before "bil_inform_bill" for the query case similar to the one in Table 5.5. All the trigger words are listed in Appendix G.

If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK
then FIRST_RESPONSE_TYPE = thank ..... (1)

If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK
AND $DA_{Customer,t}$ = REQUEST_ACTION
then NEXT_RESPONSE_TYPE = bil_inform_bill .....(2)

Figure 5.5: Hand-defined rules for generating response trigger words for the waiter responses.

In our system, we have hand-defined 125 response trigger words. Examples are shown in Table 5.6, where the word after "type" in the frame is the corresponding response trigger word. Note that some response frames need domain knowledge. As an example, in order to inform the bill to the customer, the waiter should have the amount of the bill. In our approach, the system detects the variable

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}
Waiter1	**Generated TG of the system:** BILL **Generated DA of the system:** THANK **Generated response trigger words:** thank, bil_inform_bill (The corresponding rules: If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK then FIRST_RESPONSE_TYPE = thank If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK AND $DA_{Customer,t}$ = REQUEST_ACTION then NEXT_RESPONSE_TYPE = bil_inform_bill)

Table 5.5: Continuing in the example in Table 5.1.2, the response trigger words are generated for the waiter response.

with the words beginning with "#" (e.g. #Bill_Value) and fills the values of the corresponding knowledge. Continuing in the early example, Table 5.7 shows the corresponding response frames for the customer query *"Let me have the bill, please."*, where the amount of the bill is two hundred dollars.

Response Trigger Words	Response Frames
bil_inform_bill	<response type=bil_inform_bill> <PriceValue>#Bill_Value</PriceValue> </response>
thank	<response type=thank></response>

Table 5.6: Hand-defined response trigger words and the corresponding frames for waiter responses.

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}
Waiter1	**Generated TG of the system:** BILL **Generated DA of the system:** THANK **Generated response trigger words:** thank, bil_inform_bill (The corresponding rules: If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK then FIRST_RESPONSE_TYPE = thank If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK AND $DA_{Customer,t}$ = REQUEST_ACTION then NEXT_RESPONSE_TYPE = bil_inform_bill) **Generated response frames:** { <response type=thank></response> <response type=bil_inform_bill> <PriceValue>two hundred dollars</PriceValue> </response> }

Table 5.7: Continuing in the example in Table 5.2, the response frames are generated for the waiter response.

5.1.4 Text Generation

Text generation generates text responses from response frames. We have a similar implementation as the cooperative response generation process. Each response trigger word maps to a corresponding text response template. Here, we have hand-defined 125 rules for mapping the response frames to the text responses, illustrated by the one shown in Table 5.8. The text response is generated by mapping with the response trigger word. The "#" variables are filled with values from the response frames. In order to make the system more interesting, we implemented more than one text response for some response frames. For example, the response trigger word "thank" can be mapped to text response "Thank you.", "Thank you very much." and "Thanks". In our system, we randomly pick one of the text responses with the same response trigger word in each turn. Table 5.9 shows the text response generated for the customer request "Let me have the bill, please.".

Trigger Word	Text responses
greeting	Hi. / Hello.
welcome_phrase	Welcome to #Restaurant.
offer_help	Can I help you, sir? / Can I help you? / What can I help you? / May I help you?
bil_inform_bill	Your bill goes to #Bill_Value, please. / It's #Bill_Value, sir.
thank	Thank you. / Thank you very much. / Thanks.

Table 5.8: Mapping of response trigger words to text response for text generation.

5.2 Experiments and Results

We have designed an evaluation test to assess the effectiveness of our dialog system. An evaluation test is set up with 20 subjects. Each subject is asked to follow the tasks defined in the questionnaire and interact with the dialog system via text

Customer1	*"Let me have the bill, please."* **Task Goal (TG):** BILL **Dialog Act (DA):** REQUEST_ACTION **State Space Representation:** {BILL, REQUEST_ACTION}
Waiter1	**Generated TG of the system:** BILL **Generated DA of the system:** THANK **Generated response trigger words:** thank, bil_inform_bill (The corresponding rules: If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK then FIRST_RESPONSE_TYPE = thank If $TG_{Waiter,t}$ = BILL AND $DA_{Waiter,t}$ = THANK AND $DA_{Customer,t}$ = REQUEST_ACTION then NEXT_RESPONSE_TYPE = bil_inform_bill) **Generated cooperative response frames:** { <response type=thank></response> <response type=bil_inform_bill> <PriceValue>two hundred dollars</PriceValue> </response> } **Text response:** Thank you. Your bill comes to two hundred dollars, please.

Table 5.9: Continuing in the example in Table 5.2, the corresponding text response is generated.

input. The task scenarios are shown in Table 5.10. During the interaction with the system, each subject pretends to be a customer in a restaurant while the system acts as a virtual waiter. For every task scenario, the subject is required to answer 5 questions on a scale of 1 to 5 (1 represents very bad, and 5 represents very good). Explanations and demonstrations are given to the subjects before the tests, to ensure that they have enough knowledge for the judgement. The questionnaire is shown in Appendix H.

The evaluation test of our dialog system focuses on both the *objective* or *subjective* measures. *Objective metrics* can be calculated without human judgement, while *subjective metrics* require subjects to evaluate and categorize the dialogs in various qualitative dimensions. Details are illustrated in the following sub-sections.

Background	You are Mr. Chan. You are now in a restaurant talking with a virtual waiter. This virtual waiter can serve you anything in the restaurant domain, such as restaurant reservations, food ordering, billing, complaints and information answering. You are now trying to communicate with him in order to complete the following tasks.
Reservation Task	You want to reserve a table with the following details: (i) Time of Reservation: 7:30 pm tomorrow (ii) Number of people: five (iii) Location of the table you want: by the window (iv) Your telephone number: 96112341
Food Ordering Task	You are now trying to order one smoked turkey with green salad and a cup of Chinese tea in the restaurant.
Billing Task	You want to have a bill now.

Table 5.10: Three task scenarios on the questionnaire.

5.2.1 Subjective Results

At the end of the test, each user provided a subjective evaluation of the system's performance for each task scenario via a questionnaire. For each task scenario, the questionnaire elicits perceived user satisfaction and metrics based on Grice's maxims, as shown in Table 5.11. Users reported their perceptions via three sets of 5 survey questions as shown in Figure 5.6. Each of the 5 questions for a task scenario stated one maxim for evaluation. All questions in the questionnaire were stated positivly and the user is asked to specify the degree ranging from 1 to 5 to which they agree with these statements. We then formulated a *t-test* using the difference in these opinion scores as our test statistics. In the tests, we compare each subjective metric with the score of 3, the midpoint of our pre-set degree range and treated as "average". Results are shown in Table 5.12. Testing at a significance level of 0.05 concludes that we should accept the alternate hypothesis. That is, the users believe that our system provides better performance than average, in terms of Grice's maxims and user satisfaction for each task scenario. The details of the statistical test are shown in Appendix I.

Maxim	Explanation
Maxim of Quality	System responses should be true with adequate evidence
Maxim of Quantity	System should give sufficient information
Maxim of Relevance	System responses should be relevant to the ongoing conversation
Maxim of Manner	System responses should be brief and clear, with no obscurity or ambiguity

Table 5.11: Grice's maxims and the corresponding definitions.

1. Do you think the answers of the virtual waiter are relevant to your questions? (Maxim of Relevance)
2. Do you think the answers of the virtual waiter are clear? (Maxim of Manner)
3. Do you think the virtual waiter has made true statements? (Maxim of Quality)
4. Do you think the answers of the virtual waiter are informative as you expected? (Maxim of Quantity)
5. In overall, do you satisfy with the performance of the dialog manager for this task? (Perceived User Satisfaction)

Figure 5.6: Survey questions for assessing perceived user satisfaction in each task scenario.

		Maxim of Relevance	Maxim of Manner	Maxim of Quality	Maxim of Quantity	Perceived User Satisfaction
RESERVATION (Task One)	$\bar{x}$	3.35	3.45	3.45	3.55	3.35
	s	0.81	0.83	1.05	0.83	0.81
ORDER (Task Two)	$\bar{x}$	3.9	3.8	3.9	3.9	3.8
	s	0.55	0.77	0.85	0.72	0.89
BILL (Task Three)	$\bar{x}$	3.5	3.55	3.5	3.55	3.5
	s	1.19	1.23	1.19	1.23	1.19

Table 5.12: The statistics of our opinion scores for each task scenario in terms of the range from 1 to 5, where the score 5 represents very good and the score 1 represents very poor. Here, $\bar{x}$ indicates the sample mean while s indicates the sample standard deviation.

5.2.2 Objective Results

We examined the task completion rate by scenarios. In the questionnaire, three task scenarios were given to the users. Each user was asked to follow and complete the scenarios as shown in Table 5.10. Then, after all the subjects have finished the evaluation test, we checked the system log file and detected the number of dialog turns and the task completion for each subject. The total task completion rate and the average number of dialog turns for each scenario are shown in Table 5.13. In error analysis, we found that the users failed in using the system because of insuffiency in our category tagging and incorrect goal identification; hence our system cannot understand what the user requests. For example in Table 5.14, the word "only me" in the customer query (Customer2) cannot be captured in our category tagging, but in fact this query indicates the user requests a table for one only. The system continues the dialog until the user is forced to say "one" in the query (Customer3).

	RESERVATION (Task One)	ORDER (Task Two)	BILL (Task Three)
Task Completion Rate	90% (18/20)	95% (19/20)	95% (19/20)
Average Number of Dialog Turns	4.4	2.7	2.8

Table 5.13: The Results of task completion rate and the average number of dialog turns of each task scenario.

5.3 Chapter Summary

This paper describes an implementation model, which generates cooperative responses in the CU Restaurants domain based on the understanding of the interdependencies among dialog acts, task goals and discourse inheritance. Evaluation

Waiter1:	*"Welcome to HCCL Restaurant."* **Categories:** { <Welcome_Phrase = "welcome"> <Restaurant = "HCCL Restaurant"> } **TG:** RESERVATION **DA:** GREET
Waiter1:	*"For how many persons, please?"* **Categories:** { <How_Many = "how many"> <People = "persons"> } **TG:** RESERVATION **DA:** REQUEST_INFO
Customer2:	*"Only me."* **Categories:** { NIL} **TG:** RESERVATION **DA:** INFORM
Waiter2:	*"For how many persons, please?"* **Categories:** { <How_Many = "how many"> <People = "persons"> } **TG:** RESERVATION **DA:** REQUEST_INFO
Customer3:	*"One."* **Categories:** { <NumberValue = "one"> <People = "persons"> } **TG:** RESERVATION **DA:** INFORM
Waiter3:	*"I'm afraid that we are full right now..."*

Table 5.14: An example dialog extracted from the log file indicating an insufficiency in our category tagging.

includes both subjective and objective metrics, which indicates that our system achieves a promising result on both user satifiaction and task completion rate.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we have investigated different representational and computational frameworks to model mixed-initiative dialogs. We first begin with a system known as ISIS, which abbreviates as *Intelligent Speech for Information Systems.* ISIS is a trilingual system, supporting English, Cantonese and Putonghua - the predominant languages used in our region. The system resembles a virtual stockbroker, which can provide the user with real-time stock market information and personal portfolio information. It also handles simulated financial transactions. In the flow of the conversation, the system can prompt for missing information, offer price alert services, invoke confirmation sub-dialogs, inherit information from context, reject out-of-domain queries and generate time-out messages to the user. Besides, ISIS has a list of meta-commands e.g. *help*, *good-bye*, *undo*, etc., to allow the user to navigate freely in the dialog space. In ISIS, discourse inheritance is achieved by using a form-based approach and inherits the values of the *missing* attributes *selectively* from the discourse to the current user input. ISIS targets for the financial domain, which provides two complexities in dialog management, including the

newly listed stocks problem and the interaction involving multiple tasks. We have begun on two new directions within the ISIS context. The first is to explore the development of a learning system, where the system's knowledge base is automatically expanded through typed interaction with the user. In the work on "learning sub-dialogs", we focus on text input, bypassing the issues that have to do with misrecognition. The second is to explore transitions between online interaction and offline delegation in a single dialog thread. Hence, ISIS can allow the user to switch freely between these two tasks. In our work, this mechanism exists in both spoken and text-based interactions.

In the second part of this thesis, we have studied the interdependencies among dialog acts, task goals and discourse inheritance for mixed-initiative dialogs in the CU Restaurants domain. Our study is based on 199 dialogs in the restaurants domain, with disjoint training (169 dialogs) and test sets (30 dialogs). Our training set is first annotated manually in terms of task goals and dialog acts, and tagged automatically in terms of semantic and syntactic categories for each customer request and waiter response. Based on the analysis of the annotation process, we have written a set of category inheritance and refresh rules, which constitute our selective inheritance strategy. We used Belief Networks (BNs) to automate identification of task goals and dialog acts based on input categories. Selective category inheritance outperformed two alternate control strategies where none or all of the categories are inherited. Our evaluation experiments are conducted by using text-based customer queries, bypassing the issues that have to do with misrecognition. The selective strategy achieved correct task goal identification for 92.6% of the dialog turns and correct dialog act identification for 97.8% of the text-based utterances in the test set. We have also developed a discourse inheritance procedure, which can correctly handle 95.9% of the dialog turns in the test set.

The categories, task goal(s) and dialog act(s) from the request should be useful

for the automatic generation of a coherent response. Aside from the automatic analysis of the user's requests, we should study how to respond in a mixed-initiative fashion. Hence in the third part of this thesis, we have leveraged from our model of discourse inheritance in the CU Restaurants domain to a rule-based dialog system, which supports text-based interactions, and generates cooperative response based on the identified task goals, dialog acts and categories. In the application, the user acts as a customer and gives requests via text input. The system resembles a virtual waiter, which can handle inquiries about billing, food ordering, table reservation, complaints and other services, such as requesting extra utensils or menu and the information about the restaurant. We have also designed an evaluation test to assess the effectiveness of our dialog system. An evaluation test is set up with 20 subjects. Each subject is asked to follow the tasks in a questionnaire to evaluate our dialog system. Our evaluation design has been focused on both *objective* and *subjective* metrics. *Objective metrics* can be calculated without human judgment, while *subjective metrics* require subjects to evaluate and categorize the dialogs in various qualitative dimensions. In our approach, we calculated the percentage of task completion as our objective metrics and we followed Grice's maxims in our subjective metric consideration. Evaluation indicated that our system achieves a promising result on both user satisfaction and task completion.

6.2 Contributions

In this thesis, the following contributions are made to the research area of mixed-initiative dialog modeling:

1. Within the context of the spoken dialog system ISIS, we have designed and implemented a mixed-initiative dialog model capable of

    - inheriting selected discourse history,

- combining online interaction with offline delegation sub-dialogs, and
- learning new information through a conversation to expand the system's knowledge space.

2. From the study of the interdependencies among dialog acts, task goals and discourse inheritance in mixed-initiative human-human dialogs in the CU Restaurants domain, we observed the following [80]:

   - The task goal of a context-independent customer request can be identified from its categories, while the task goal of a context-dependent request often may require inheritance from the previous dialog turn.
   - Dialog act identification does not require category inheritance, while task goal identification does.
   - Category inheritance is largely dependent on the task goals of the current query. However, in some cases it is also influenced by the dialog act in the case of confirmation or rejection.

3. Based on the results of our observations and analysis from the study, we have developed a discourse inheritance procedure for handling customer requests in the CU Restaurants domain.

4. An implementation model is developed for generating cooperative responses based on our analysis results in the CU Restaurants domain.

6.3 Future Work

Possible extensions of this work include:

1. **Incorporating anaphora resolution in our dialog and discourse.** Anaphora resolution is a key issue in Natural Language Processing (NLP)

and a number of other NLP applications. In our current dialog model in both ISIS and CU Restaurants domain, we have not investigated any problem regarding the choices of referring expressions in discourse inheritance. With the help of anaphora resolution, we can know more about the discourse structures of the dialogs. This should enable us to improve our selective discourse inheritance strategies in both domains.

2. **Integrating the dialog act consideration in the dialog model of ISIS.** In ISIS, the initiative control depends on just the task goal. In order to make the system more intelligible, we can integrate dialog act consideration into the ISIS domain. We may leverage our observations and experience from the analysis in the CU Restaurants domain to the ISIS domain.

3. **Developing an automatic dialog generator.** Since it depends on handcrafting for a sophisticated dialog flow, the process is expensive. In the study of the CU Restaurants domain, our dialog corpus is collected from websites and books for English learning. Hence the source of training and testing dialogs are all in text format. In our rule-based cooperative response system, the rules are hand-defined based on different waiter responses in the training set. In order to reduce the amount of handcrafting, it should be desirable to get all the rules which are generated automatically by a program or a system. Extending this idea, it should be desirable, if there is a system which uses some dialog transcripts as input and generates all the rules regarding the dialog flow and cooperative responses. In this task, a machine-learning approach should help in developing such an automatic dialog generator.

4. **Extending our dialog system in the CU Restaurants domain to handle spoken customer utterances.** Our dialog system in the the CU Restaurants domain can only receive customer requests via text input.

In order to make our system more sophisticated, we should apply more speech and language technologies with our dialog modeling techniques. For example, we can integrate speech recognition techniques, so that our virtual waiter can hear the spoken requests of the customer; and we can integrate speech generation techniques so that our virtual waiter can respond to the customer in speech.

5. **Considering emotions from spoken customer utterances.** While extending our dialog system in the CU Restaurants domain to handle spoken customer utterances, we can further do some research related to emotion identification. Emotions can be recognized by analyzing the utterances of the user in many aspects, e.g. vocabulary and prosody. Depending on these emotions, the dialog system may respond more effectively, for example, by giving the user help if he or she feels annoyed.

Bibliography

[1] W. Eckert, E. Levin, and R. Pieraccini, User modeling for spoken dialogue system evaluation, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

[2] J. R. Glass, Challenges for spoken dialogue systems, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[3] E. Levin and R. Pieraccini, Spoken language dialogue: From theory to practice, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[4] L. P. Kaelbling and A. W. Moore, Reinforcement learning: A survey, in *Journal of Artifical Intelligence Research 4 237-285*, 1996.

[5] H. Meng, W. Lam, and K. F. Low, Learning belief networks for language understanding, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[6] H. Meng et al., Isis: A learning system with combined interaction and delegation dialogs, in *Proceedings of European Conference on Speech Communication and Technology*, 2001.

[7] J. Alexandersson et al., Dialog acts in verbmobil-2 second edition, in *Verbmobil Report 226, Universitat Hamburg, DFKI Saarbrucken, Universitat Erlangen, TU Berlin*, 1998.

[8] M. A. Walker and S. Whittaker, Mixed initiative in dialogue: An investigation into discourse segmentation, in *Meeting of the Association for Computational Linguistics*, pages 70–78, 1990.

[9] S. Whittaker and P. Stenton, Cues and control in expert-client dialogues, in *Proccedings of ACL*, pages 123–130, 1988.

[10] V. Zue, Conversational interfaces: Advances and challenges, in *Proceedings of European Conference on Speech Communication and Technology*, 1997.

[11] J. Allen, Mixed initiative interaction, in *Proceedings of IEEE Intelligent Systems*, 1999.

[12] R. W. Smith, *A Computational Model of Expectation Driven Mixed Initiative Dialog Processing*, PhD dissertation, Duke University, 1992.

[13] H. Meng et al., WHEELS: A conversational system in the automobile classifieds domain, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 542–545, Philadelphia, PA, 1996.

[14] R. Cohen et al., User Modeling and User-Adapted Interaction **vol 8**, page 171 (1998).

[15] C. I. Guinn, Evaluating mixed-initative dialog, in *Proceedings of IEEE Intelligent Systems*, 1999.

[16] M. H. Burstein and D. V. McDermott, Cognitive Technology: In Search of a Humane Interface Edited by B. Gorayska and J. L. Mey. Elserver Science B. V. (1996).

[17] C. Wang et al., Muxing: A telephone-access mandarin conversational system, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2000.

[18] V. Zue et al., Jupiter: A telephone based conversational interface for weather information, in *Proceedings of IEEE Transactions on Speech and Audio Processing*, 2000.

[19] S. Seneff and J. Polifroni, Dialogue management in the mercury flight reservation system, in *Proceedings of the Satellite Dialogue Workshop, ANLP-NAACL*, 2000.

[20] V. Zue et al., Speech Communication , page 331 (1994).

[21] R. Carlson, Recent developments in the experimental waxholm dialog system, in *ARPA Human Language Technology Workshop*, 1994.

[22] C. Huang et al., Lodestar: A mandarin spoken dialogue system for travel information retrieval, in *Proceedings of European Conference on Speech Communication and Technology*, 1999.

[23] S. Bennacef, L. Devillers, S. Rosset, and L. Lamel, Dialog in the railtel telephone-based system, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

[24] L. Lamel, S. Bennacef, J. L. G. ans H. Dartigues, and J. N. Temem, User evaluation of the mask kiosk, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1998.

[25] J. Sturm, E. Os, and L. Boves, Dialogue management in the dutch arise train timetable information system, in *Proceedings of European Conference on Speech Communication and Technology*, 1999.

[26] H. Meng et al., Isis: A multilingual spoken dialog system developed with corba and kqml agents, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2000.

[27] H. Meng, S. Lee, and C. Wai, Cuforex: A bilingual spoken dialog system for foreign exchange enquires, in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2000.

[28] D. Jurafsky, C. Wooters, and G. Tajchman, The berkeley restaurant project, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1994.

[29] M. Aretoulaki et al., Sqel: A multilingual and multifunctional dialogue system, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1998.

[30] S. Seneff et al., Galaxy-ii: A reference architecture for conversational system development, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1998.

[31] K. Kita, Y. Fukui, M. Nagata, and T. Morimoto, Automatic acquisition of probabilistic dialogue models, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

[32] B. L. Zeigler and B. Mazor, Dialog design for a speech-interactive automation system, in *Proceedings of the 2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA)*, 1994.

[33] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, A form-based dialog manager for spoken langauge applications, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

[34] K. A. Papineni, S. Roukos, and R. T. Ward, Free flow dialog management using forms, in *Proceedings of European Conference on Speech Communication and Technology*, 1999.

[35] B. Pellom, W. Ward, and S. Pradhan, The cu communicator: An architecture for dialogue systems, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2000.

[36] W. Ward and B. Pellom, The cu communicator system, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[37] S. Bayer, C. Doran, and B. George, Exploring speech-enabled dialog with the galaxy communicator infrastructure, in *Proceedings of the Human Language Technology Conference*, 2001.

[38] R. Pieraccini, E. Levin, and W. Eckert, Amica: the at&t mixed initiative conversational architecture, in *Proceedings of European Conference on Speech Communication and Technology*, 1997.

[39] E. Levin et al., The at&t-darpa communicator mixed-initiative spoken dialog system, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

[40] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[41] N. J. Nilsson, Introduction to machine learning, in *Robotics Laboratory, Department of Computer Science, Standford University, Standford, CA 94305*, 1996.

[42] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, N.J., 1995.

[43] B. S. Lin and L. S. Lee, Computer-aided analysis and design for spoken dialogue systems based on quantitative simulations, in *IEEE Transactions on Speech and Audio Processing*, 2001.

[44] E. Levin, R. Pieraccini, and W. Eckert, A stochastic model of human-machine interaction for learning dialog strategies, in *IEEE Transactions on Speech and Audio Processing*, 2000.

[45] D. Goddeau and J. Pineau, Fast reinforcement learning of dialogue strategies, in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2000.

[46] D. J. Litman, M. S. Kearns, S. Singh, and M. A. Walker, Automatic optimization of dialogue management, in *Proceedings of COLING*, 2000.

[47] M. A. Walker, J. Fromer, and S. Narayanan, Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email, in *Proceedings of ACL/COLING*, 1998.

[48] E. Levin and R. Pieraccini, A stochastic model of computer-human interaction for learning dialogue strategies, in *Proceedings of European Conference on Speech Communication and Technology*, 1997.

[49] D. Heckerman, A tutorial on learning with bayesian networks, in *Technical Report, MSR-TR-95-06, MicroSoft Research*, 1996.

[50] T. Inamura, M. Inaba, and H. Inoue, Integration model of learning mechanism and dialogue strategy based on stochastic experience representation using bayesian network, in *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*, 2000.

[51] F. V. Jensen, *An Introduction to Bayesian Networks*, 1996.

[52] H. Meng, C. Wai, and R. Pierracinni, The use of belief networks for mixed-initiative dialog modeling, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Beijing, China, 2000.

[53] J. Pearl, *Bayesian Networks*, Technical Report (R-216), Revision I In M. Arbib(Ed.), Handbook of Brain Theory and Neural Networks, MIT Press, 149-153, 1995.

[54] H. Meng, W. Lam, and C. Wai, To believe is to understand, in *Proceedings of European Conference on Speech Communication and Technology*, 1999.

[55] C. Doran, J. Aberdeen, L. Damianos, and L. Hirschman, Comparing several aspects of human-computer and human-human dialogs, in *Proceedings of the 2nd Sigdial Workshop on Discourse and Dialog, Aalborg*, 2001.

[56] M. Walker et al., Darpa communicator dialog travel planning systems: The june 2000 data collection, in *Proceedings of European Conference on Speech Communication and Technology*, 2001.

[57] M. Walker and R. Passonneau, Date: A dialog act tagging scheme for evaluation of spoken dialog systems, in *Proceedings of the 2nd Sigdial Workshop on Discourse and Dialog, Aalborg*, 2001.

[58] Cstar consortium, in *Dialog act annotation, Unpublished Manuscript*, 1999.

[59] R. Frederking, Grice's maxims: do the right thing, in *Frederking, R. E. (1996). Grice's maxims: do the right thing. In Working Notes of the AAAI96 Spring Symposium on Computational Approaches to Interpreting and Generating Conversational Implicature, pages 21–26, Menlo Park, CA. AAAI Press.*, 1996.

[60] H. P. Grice, Logic and conversation, in *P. Cole and J.L. Morgan, editors, Syntax and semantics, volume 3, pages 41–58. Academic Press, New York*, 1974.

[61] P. G. T. Gaasterland and J. Minker, An overview of cooperative answering, in *Journal of Intelligent Information Systems, Kluwer Academic Publishers, vol. 1, no. 2, pp. 123-157*, 1992.

[62] J. Chu-Caroll, Response generation in collaborative dialogue interactions, in *Proceedings the 33rd ACL, pages 136-143*, 1995.

[63] J. Chu-Carroll and S. Carberry, A plan-based model for response generation in collaborative task-oriented dialogues, in *Proceedings of the 12th AAAI, pages 799-805*, 1994.

[64] Y. Qu and S. Beale, A constraint-based model for cooperative response generation in information dialogues, in *In Proceedings of AAAI-99, Orlando, FL*, 1999.

[65] H. Meng et al., Isis: A trilingual conversational system with learning capabilities and combined interaction and delegation dialogs, in *Proceedings of the National Conference on Man-Machine Speech Communication (NCMMSC6), Shenzhen, November*, 2001.

[66] H. Meng, P. C. Ching, Y. F. Wong, and C. C. Chan, Isis: A multi-modal, trilingual, distributed spoken dialog system developed with corba, java, xml and kqml, in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

[67] H. Meng and W. C. Tsui, Comprehension across domains and languages, in *Proceedings of the International Symposium of Chinese Spoken Language Processing (ISCSLP)*, 2000.

[68] G. Damnati and F. Panaget, Adding new words in a spoken dialogue system vocabulary using conceptual information and derived class-based lm, in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[69] E. Brill, Automatic grammar induction and parsing free text a transformation-based approach, in *Proccedings of ACL*, 1993.

[70] 鄭慧思, 各行各業合售貨員英語手册, 通用語言唱片公司, 2001.

[71] 林志輝, 酒樓餐廳英語, 星輝圖書有限公司, 1996.

[72] 商務編輯部, 速成英語會話, 精英, 2001.

[73] 史濟蘭, 英語會話入門, 學習出版社有限公司, 1996.

[74] S. Jekat et al., Dialog acts in verbmobil, in *Verbmobil Report 65, Universitat Hamburg, DFKI Saarbrucken, Universitat Erlangen, TU Berlin*, 1995.

[75] D. Jurafsky et al., Automatic detection of discourse structure for speech recognition and understanding, in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, 1997.

[76] D. Jurafsky, E. Shriberg, and D. Biasca, Switchboard swbd-damsl shallow-discourse-function annotation coders manual, draft 13, in *University of Colorado, Boulder, Institute of Cognitive Science Technical Report*, 1997.

[77] A. Stolcke et al., Dialog act modelling for automatic tagging and recognition of conversational speech, in *Computational Linguistics, 26:3, 339-371*, 2000.

[78] M. Walker, R. P. R., and J. Boland, Quantitative and qualitative evaluation of darpa communicator spoken dialog systems, in *Proceedings of the Human Language Technology Conference*, 2001.

[79] C. Doran, J. Aberdeen, L. Walker, and B. Passonneau, Guidelines for tagging dialog acts in human-human and human-computer travel dialogs, 2001.

[80] S. F. Chan and H. Meng, Interdependencies among dialog acts, task goals and discourse inheritance in mixed-initiative dialogs, in *Proceedings of the Human Language Technology Conference*, 2002.

Appendix A

Domain-Specific Task Goals in CU Restaurants Domain

ASK_INFO:	
Definitions:	The utterance contains a content on asking information.
Example:	"Where is the telephone?"
BILL:	
Definitions:	The utterance contains a content related to billing.
Example:	"I would like to bill please.", "How much is it?"
COMPLAINT:	
Definitions:	The utterance contains a content on complaint.
Example:	"I would like to make a complaint.", "The steak is under-cooked!"
ORDER:	
Definitions:	The utterance contains a content on ordering food.
Example:	"What would you recommend for desserts?", "I would like to have an American breakfast."
RESERVATION:	
Definitions:	The utterance contains a content on requesting a table.
Example:	"A table for four please.", "Can I have a table?"
SERVING:	
Definitions:	The utterance contains a content on serving.
Example:	"Can you bring me some matches please?", "Can I have some toothpick please?"

Table A.1: Definitions and examples of 6 task goals corresponding to the CU Restaurants domain.

Appendix B

Full list of VERBMOBIL-2 Dialog Acts

Accept	Backchannel	Bye
Clarify	Close	Commit
Confirm	Defer	Deliberate
Deviate Scenario	Explained Reject	Digress
Exclude	Feedback	Feedback Negative
Feedback Positive	Give Reason	Greet
Inform	Init	Introduce
Not Classifiable	Offer	Politeness Formula
Refer to Setting	Reject	Request
Request Clarify	Request Comment	Request Commit
Request Suggest	Suggest	Thank

Table B.1: Full list of VERBMOBIL-2 dialog acts.

Appendix C

Dialog Acts for Customer Requests and Waiter Responses in CU Restaurants Domain

BACKCHANNEL:	
Definitions:	With a BACKCHANNEL the customer solely signals that he is still following the conversation.
Example:	"I see.", "Well...", "Hmm..."
CLOSE:	
Definitions:	With a CLOSE the customer says good bye to the waiter, thereby closing the dialog.
Example:	"Bye!", "See you tonight", "Good-bye"
DEFER:	
Definitions:	The customer explicitly suggests or announces the interruption of the topic currently dealt with in the dialog.
Example:	"Let's see.", "Can you give me a few seconds?"
FEEDBACK_NEGATIVE:	
Definitions:	With an utterance expressing FEEDBACK_NEGATIVE the customer reacts to a contribution of the dialog partner in a negative way. A FEEDBACK_NEGATIVE can signal rejection or dislikes of the contents of a previous contribution or it can express a negative answer to a yes/no question.
Example:	"No.", "That's a nuisance."

FEEDBACK_POSITIVE:	
Definitions:	With an utterance expressing FEEDBACK_POSITIVE the customer reacts to a contribution of the dialog in a positive way. A FEEDBACK_POSITIVE can signal acceptance of the content of a previous contribution or it can express a positive answer to a yes/no question.
Example:	"Yes.", "OK", "That's good", "Fine"
GREET:	
Definitions:	GREET is used for all kinds of initial greetings.
Example:	"Hello!", "Good morning.", "Hi."
INFORM:	
Definitions:	The label INFORM is reserved for cases where none of the categories apply. If not enough information is available in the content to label the given dialog segment as any of those it can be labeled as INFORM.
Example:	"Here it is.", "I ordered dinner about half an hour ago."
PREFER:	
Definitions:	With a PREFER the customer signals his/her preference on the content of previous conversation.
Example:	"I'll have this.", "I would like to have a seafood platter, please."
REQUEST_ACTION:	
Definitions:	The customer explicitly requests to perform one or more specified actions (e.g. handling complaints, ordering food, making a reservation, looking for information).
Example:	"Can you arrange a dinner for me?", "I would like to make a reservation."
REQUEST_COMMENT:	
Definitions:	With an utterance expressing a REQUEST_COMMENT the customer explicitly asks his waiter to comment on a proposal. It is often used to yield the turn; in that case it prompts the dialog partner to respond.
Example:	"Is your spicy pasta really hot?", "Is it good?"
REQUEST_INFO:	
Definitions:	With an utterance expressing a REQUEST_INFO the customer asks the dialog partner to present information or more information about something that has already been either explicitly or implicitly introduced into the discourse.
Example:	"How much is the voucher worth?", "Where is the telephone?"

REQUEST_SUGGEST:	
Definitions:	With an utterance expressing a REQUEST_SUGGEST the customer asks the dialog partner to make a suggestion or proposal.
Example:	"What would you recommend?", "What do you recommend then?"
SUGGEST:	
Definitions:	With an utterance SUGGEST expressing a suggest the customer proposes an explicit instance or aspect of the negotiated topic.
Example:	"Can't you pull two tables together?", "How about make it for two only?"
THANK:	
Definitions:	With an utterance expressing a THANK the customer expresses his gratitude to the dialog partner.
Example:	"Thank you", "Thank you very much"

Table C.1: Definitions and examples of 14 dialog acts for customer requests in CU Restaurants domain.

APOLOGY:	
Definitions:	With an APOLOGY the waiter solely signals regret to customer.
Example:	"I'm sorry, sir."
BACKCHANNEL:	
Definitions:	With a BACKCHANNEL the waiter solely signals that he is still following the conversation.
Example:	"I see.", "Well...", "Hmm..."
CLOSE:	
Definitions:	With a CLOSE the waiter says good bye or closes the conversation to the customer.
Example:	"Good-bye, have a nice day.", "Please come again."
COMMIT:	
Definitions:	With a COMMIT the waiter explicitly commits him/herself to do one or more specific actions to the customer.
Example:	"I'll show you the new table.", "We can seat you very soon."
CONFIRM:	
Definitions:	The waiter wraps up the result of the negotiation to the customer.
Example:	"You have ordered one apple juice, a chicken salad and a hamburger."

DEFER:	
Definitions:	The customer explicitly suggests or announces the interruption of the topic currently dealt with in the dialog.
Example:	"Let's see.", "Can you give me a few seconds?"
FEEDBACK_NEGATIVE:	
Definitions:	With an utterance expressing FEEDBACK_NEGATIVE the waiter reacts to a contribution of the dialog partner in a negative way. A FEEDBACK_NEGATIVE can signal rejection or dislikes of the contents of a previous contribution or it can express a negative answer to a yes/no question.
Example:	"No, sir.", "We cannot change the portions from four people to 2 people for you."
FEEDBACK_POSITIVE:	
Definitions:	With an utterance expressing FEEDBACK_POSITIVE the waiter reacts to a contribution of the dialog in a positive way. A FEEDBACK_POSITIVE can signal acceptance of the content of a previous contribution or it can express a positive answer to a yes/no question.
Example:	"Yes.", "OK", "That's good", "Great"
GREET:	
Definitions:	GREET is used for all kinds of initial greetings.
Example:	"Hello!", "Good morning.", "Hi."
INFORM:	
Definitions:	The label INFORM is reserved for cases where none of the categories apply. If not enough information is available in the content to label the given dialog segment as any of those it can be labeled as INFORM.
Example:	"Here it is.", "There are fresh strawberries for dessert."
INTRODUCE:	
Definitions:	The utterance contains information about the speaker, e.g. his/her name, title, position or profession.
Example:	"I'm John. I will be your server tonight."
OFFER:	
Definitions:	The waiter explicitly offers to perform one or more specific actions.
Example:	"May I help you?", "Can I give you the menu, sir?"

REQUEST_COMMENT:	
Definitions:	With an utterance expressing a REQUEST_COMMENT the waiter explicitly asks his dialog partner to comment on a proposal. It is often used to yield the turn; in that case it prompts the dialog partner to respond.
Example:	"How is the wine?", "Is it good?"
REQUEST_INFO:	
Definitions:	With an utterance expressing a THANK the customer expresses his gratitude to the dialog partner.
Example:	"You mean the public phone?", "Anything else?"
SUGGEST:	
Definitions:	With an utterance SUGGEST expressing a suggest the speaker proposes an explicit instance or aspect of the negotiated topic.
Example:	"I would recommend the Roses.". "How about the steamed tofu in oil?"
THANK:	
Definitions:	With an utterance expressing a THANK the customer expresses his gratitude to the dialog partner.
Example:	"Thank you, sir."

Table C.2: Definitions and examples of 16 dialog acts for waiter responses in CU Restaurants domain.

Appendix D

The Two Grammers for Task Goal and Dialog Act Identification

Amount: amount, amounts, quantities, quantity, volume, volumes, portions, portion
Appointment appointment, appointments, booking, bookings
Arrange arrange, arranging, arranged, arranges
Arrive arrived, arrive, arrives, arriving
Bill bill, bills, billing
BreadStyle whole wheat, rye, plain, dark
Call call, calls, calling, called

Cancel
cancel, skip, delete, cancelled, cancelling, skipping, skipped, deleted, deleting
Care
careful
Changes
changes, right change, right changes
Chef
chef, the cook
Chew
chew, chews, chewed, chewing
CHUNK
For all unseen words
Come
come, coming, comes
Complain
complain, complained, complaint, complaining
Cook
cook, cooked, cooking
CookStyle
stir fried, marinated, pan fried, oiled, steamed, baked, grilled, roast, smoked, roasted, creamed, grated, boiled, canned, filet wrapped, fried, basted, charbroiled, barbecued, iced, diced, sliced, minced, mashed, stuffed, boil
Cost
costs, costing, cost, costed, worth
Country
new york, taiwan, japanese, american, chinese, french, germany, swiss, foreign, taiwanese, scotch
Course
a course, b course, c course, d course, e course, f course, g course, course, courses, set course, set courses, set dinner, set lunch
Criticism
overcharge, overcharged, mistake, grease, nuisance, underdone, bloody, awful, long, very BadWord, too BadWord, so BadWord, take so long, so BadWord, bad, trouble, so long, cracked, too rare, too Description, too Size

DateDay

mon, tue, wed, thu, thur, fri, sat, sun, monday, tuesday, wednesday, thursday, friday, saturday, sunday, mondays, tuesdays, wednesdays, thursdays, fridays, saturdays, sundays

DateMonth

jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec, january, february, march, april, june, july, august, september, october, november, december, januaries, februaries, marches, aprils, junes, julies, augusts, septembers, octobers, novembers, decembers

Description

expensive, new, fresh, crisp, light, very Description, too Description, so Description, red, pink, white, black, substantial, interesting, high, low, glazed, sweet, subtle, mild, dry, spicy, sour, hungry, thirsty, hot, splashing, raw, quite, lovely, plain, soft, tender, deep, wonderful, charming, long, longer

Done

done

Dressing

japanese dressing, french dressing, curry dressing, blue cheese dressing, thousand island, sour cream

Drink

drink, drank, drunk, drinking

Dummy

may i, may you, may he, may she, may it, may they, may we, then, i may, you may, he may, she may, it may, they may, we may, is, i am, good, of course, thank you very much, not at all, welcome to, welcome, welcoming, in, inside, outside, into, on, from, to, about, from, under, near, we'll, they'll, he'll, she'll, it'll, you'll, i'll, it'll, they've, you've, i'm, he's, she's, they're, it's, you're, we're, i'd, she'd, he'd, you'd, we'd, it'd, here is, there is, here are, there are, that'll, that's, here's, there's, this'll, this's, these're, those're, that's, hello, howdy, hi, i, you, he, she, it, they, we, you, that, this, those, these, my, your, mine, yours, her, his, our, ours, ourselves, hers, them, their, us, me, its, myself, themselves, yourself, yourselves, herslef, himself, him, itself, a, an, the, Not Dummy

EggStyle

sunny side up, over easy, hard boiled, poached, scrambled

Else

anything else, something else, is that everything

Flavor

vanilla, chocolate, flavor, flavors, walnut

Food_Drink

fish, fried chicken, lobsters, lobster, food, coca, cola, coke, cuppucino, coffee, chinese tea, tea, coca cola, sprite, juice, Food_Item With Dressing, , brandy, bourbon, Description wine, wine, wines, aperitif, mao tai, champagne, gin, gimlet, kirsch, sherry, muscatel, burgundy, beer, beers, rose, cointreau, drambuie, chilean white wine, chilean red wine, gin fizz, screwdriver, manhattan, bloody mary, campari, martini cocktail, vermouth, white wine, red wine, fruit spritzer, garlic marinated scallops with mango salsa, smoked turkey with green salad, egg and mushroom special, green salad, grilled fish fillet with tomato herb sauce, grilled fish fillet, lobster omelette, grilled fish, fish fillet, wok fried chicken, lobster omelette, omelette with lobster, egg and mushrooms special, seafood platter, omelette, seafood platter, omelettes with lobster, egg, eggs, mushroom, mushrooms, vegetables, vegetable, roasted chicken with creamy chesse and mushroom sauce, braised pork with steamed rice, scallop with mango salsa, scallop with mango, scallops with mango, mango scallop, mango scallops, scallops with mango salsa, smoked turkey, roasted chicken, braised pork, steamed rice, seafood, omelette with lobster, egg and mushrooms special, smoke turkey with green salad, smoked turkey with green salad, smoked turkeys, roasted chickens, braised porks, steamed rice, turkey, turkeys, chicken, chickens, pork, porks, green tea ice cream, ice cream, apple, apples, strawberries, strawberry, fruit, fruits, watermelon, melon, grapefruit, guava, pineapple, pineapples, orange, oranges, grape, grapes, lemon, lemons, honey dew melon, orange juice, orange juices, pineapple juice, pineapple juices, apple juice, apple juices, tomato juice, tomato juices, gingko, nuts, nut, bamboo shoot soup, cream of corn soup, fresh ground beef steak 12 oz, t bone 16 oz, fresh ground beef, t bone, steak, steaks, vegetables, vegetable, tomato, tomatoes, potatoes, potato, onion, onions, radish, broccoli, bean, beans, leek, cauliflower, spinach, carrots, mushroom, mushrooms, cabbage, cabbages,

FoodStyle

buffet style, buffet, a la carte

Here
for here, here

How
how

HowLong
how long

HowMany
how many

HowMuch
how much

Information
information, data

Location
at the back of, in front of, in the middle of, on the top of, at the bottom of, over there, by the window, coffee shop, private room, private rooms, dining room, in the corner, near the window, at a table, at the counter, at the bar, main RESTAURANT, lobby floor, at the table, room, rooms, border, corner, hall, local, cloak room, lobby, shop, Numberth floor

Look
look, looking, looked

Make
made for, made, make, making

MealDescription
breakfast, lunch, dinner

Menu
menu, wine list

Name
name, names

Not
not, neither, cannot, aren't, isn't, no, out of season, amn't, can't, dunno, couldn't, shouldn't, won't, wouldn't, hasn't, haven't

Number
number, numbers

Numberth
first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleventh, twelfth, thirteenth, fourteenth, sixteenth, seventeenth, eighteenth, nineteenth, twentieth, thirtieth, twenty first, twenty second, twenty third, twenty fourth, twenty fifth, twenty sixth, twenty seventh, twenty eighth, twenty ninth, thirty first

NumberValue
one, two, three, four, five, six, seven, eight, nine, zero, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, NumberValue NumberValue

Open
open, opening, opened

Or
or

Order
side order, order, orders

Pay
pay, settle, paying, paid, settled

PayMethod
card, cards, credit card, credit cards, visa card, visa cards, visa, mastercard, master card, master cards, traveler checks, traveler cheque, traveler check, traveler cheques, cash, cheques, cheque, note, coin, coins, money, voucher, vouchers

Period
minute, minutes, hours, hour, second, seconds, **PeriodUnit** minute, minutes, hours, hour, second, seconds

Person
people, person, persons

PhoneNumber
telephone number, phone number, call number, my number

Prepare
prepare, prepared, preparing

Price
price, prices, pricing, priced, figure

PriceValue

MarketArea ¡dollar_sign¿ NumberValue, MarketArea NumberValue, ¡dollar_sign¿ NumberValue, NumberValue dollars, NumberValue dollars and NumberValue cents, NumberValue cents

Range

range, ranging

RelativeAmount

much, More, mini, max, little bit, half, less, little, few, a lot of, many, various, variety, a bit, quite a lot, some, minimum, maximum, fewest

RelativeDate

Numberth Of DateMonth, NumberValue DateDay RelativeDateHint, NumberValue DateMonth RelativeDateHint, Period RelativeDate-Hint, PartOfDay RelativeDate, RelativeDate PartOfDay, Period, RelativeDate PartOfDay, PartOfDay, RelativeDateHint DateDay, RelativeDateHint DateMonth, RelativeDateHint PeriodUnit, Rel-ativeDateHint SpecialPeriodUnit, RelativeDateHint RelativeDate, RelativeDate RelativeDateHint

RelativeDateHint

so far, up to now, this DateDay, this DateMonth, this PeriodUnit, this SpecialPeriodUnit, this Period, today, tomorrow, yesterday, monday, tuesday, wednesday, thursday, friday, saturday, sunday, today, tonight, next PeriodUnit, this PeriodUnit, today's, tommorrow, last, ago, moment ago, moments ago, after, past, last, previous, beginning, ending, next, today, tonight, next PeriodUnit, this PeriodUnit, today's, tomorrow, today, yesterday

RelativeTime

soon, later, moment, now, at once, NumberValue PeriodUnit, immediately, shortly, immediate, again, right now, as soon as possible, before, this afternoon, this morning, this evening, at night

Reserve

reservation, reservations, booking, appointment, appointments, reserve, reserved, booked

Restaurant

Restaurant, cafe

RestaurantName

sunny garden barbecue, floral, lion, lien an, hilton, pizza hut, regent, nan yang, farmhouse, natasha's place

Same
same, equal, identical
Seasoning
soy sauce, sesame oil, sugar, salt, pepper, lemon grass, lime juice, lime grass, olive oil, honey sause, sweet bean sauce
Separate
separate, apart, individual
Serve
serve, serving, service, pass, served, passing
Sit
sit, sat, sitting
Size
medium, large, small, smallest, largest, larger, smaller, size, big
SmokeOption
smoke, smoking, no smoking, non smoking
SpecialPeriodUnit
weekend, weekends, weekday, weekdays, Working days, Working day, holiday, holidays, public holidays, public holiday
SteakStyle
well done, medium rare, medium, rare, done
Substitute
substitute, replace, substituted, replaced, instead of
Suggest
chef's suggestion, suggest, recommend, recommendation, suggestion, suggests, advise, recommendations, suggestions, recommended, suggested,
Table
table, tables, party, parties, another table, this table, your table, that table, free places, free place
Third_Person
friend, lady, man, wife
Time
NumberValue PeriodUnit, NumberValue a ¡period¿ m ¡period¿, NumberValue p ¡period¿ m ¡period¿, NumberValue o'clock, NumberValue am, NumberValue pm

TodaySpecial special today, specials today, today's specials, today's special, today special, today specials
Try try, trying, tried
Unit piece, a piece of, pieces, slice, slices, oz, pint, a bottle of, a glass of, a plate of, a cup of, a pot of, a pitcher, a bowl
UserUtt u, u ¡colon¿, u ¡semi_colon¿
Utensil glass, bottle, glasses, bottles, warmer, coats, coat, public phone, telephone, phone, plate, spoon, fork, cup, pot, pitcher, vending machine, cloth, , plates, spoons, forks, cups, pitchers, vending machines, clothes, ashtray, toothpicks, napkin, bowl, pan, skewer, elevator, matches, chair, chairs, knife, knives, ashtrays, napkins, bowls, pans, skewers
Wait wait, waiting, just a moment, wait a moment
Waiter waiter, server, waitress, escort
What what kind of, what's, what're, what kind, what
When when, when's, when're
Where where, where's, where're
Which which, which's, which're

Table D.1: The hand-defined grammar for task goal identification.

ActionWord
take your order, take order, pull, show, dip, sprinkled, taste, settle the bill, settle my bill, have the bill, make a complaint, have a bill, cancel, take the bill, the bill, clear, get back, to complain, send up, to order, order, separate bills, arrange a dinner, bring, pass, take it back, cook, show, decent, hold the line, bill please, make a reservation, reserve, reserved, calculate, return, hurry up, make out the bill, be back, take, send
ApologyPhrase
very Apology, sorry, apologize, apology, regret
BackchannelPhrase
oh, uh, huh, um, huh uh, uh, ah, huh, hm, so, i see, hmm, well <comma>
Criticism
very BadWord, too BadWord, so BadWord, take so long, taking so long, so BadWord, bad, trouble, problem, problems, cracked, too long, so long, overcharge, overcharged, mistake, grease, nuisance, underdone, bloody, awful, delay, delayed, delaying, too dry, too expensive, too spicy, too hot, too rare, so dry, so expensive, so spicy, so hot, so rare
But
but, however, yet
ByePhrase
see you, bye, bye bye, byebye, goodbye, good bye
CHUNK
for all unseen words
Description
hot, enough
Dummy
Not forget, Pronoun, You, I, VerbToBe, NumberValue
Else
else, something else
Exclam_Mark
<exclam>

GoodWord
good, better, fine, very well, glad, very GoodWord, so GoodWord, too GoodWord, delicious, great, very GoodWord, lavish, wonderful, lovely, interesting, refreshing, appetizing, perfect, pleasure, happy, nice
GreetingPhrase
good morning, good afternoon, good evening, good night, hello, howdy, hi
Let
let's see, let, let's
NoWord
no, neither, dunno
Not
do not, don't, does not, doesn't, did not, didn't, not quite, are not, aren't, isn't, is not, amn't, am not, cannot, can't, couldn't, could not, shouldn't, won't, wouldn't, should not, will not, would not, has not, hasn't, have not, haven't, not
NumberValue
one, two, three, four, five, six, seven, eight, nine, zero, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, NumberValue NumberValue
PartOfDay
morning, evening, afternoon, night, overnight
PeriodUnit
minute, minutes, hours, hour, second, seconds
Please
please
PreferencePhrase
in the mood of, would do, let us, let me, don't mind, she'll have, he'll have, it'll have, prefer, Would like, Will like, Will have, favourite, think, decide, want, need, desire, request, prefered, thought, believed, decided, wanted, required, needed, desired, like, Pronoun Preference, Pronoun VerbToBe like to Preference
Pronoun
he, she, it, they, that, this, those, these, his, her, its, their, there

Quest_Mark
<quest>
RelativeDate
today, tonight, next day, next week, this day, this week, today's, tommorrow
RelativeTime
later, now, half <hyphen> an <hyphen> hour, half an hour, right now, as soon as possible, a few more minutes, NumberValue PeriodUnit
RequestPhrase
VerbToBe Pronoun, VerbToBe You, Not Pronoun, Not I, Not You, VerbToBe Pronoun like, VerbToBe You like
RequestCommentPhrase
are you sure, what are they like
Still
still
SuggestPhrase
can't you, why don't you, can i substitute something else, something else instead, which is GoodWord, which is good, which is BadWord, anything can be substituted, what do you have, suggestion
ThankPhrase
thanks, thank you, thank you very much, thanks a lot
Third_Person
friend, lady, man
Time
NumberValue PeriodUnit, NumberValue a <period> m <period>, NumberValue p <period> m <period>, NumberValue o'clock, NumberValue am, NumberValue pm
UserUtt
u, u <colon>, u <semi_colon>
VerbToBe
am, are, been, be, being, is, was, were, would be, will be, maybe, may be, do, can be, should be, has, have
Waiter
waiter, server

Wh_Word
how much, howmany, what, which, what, why, who, whose, where, when, how
YesWord
not that bad, yes, yeah, certainly, sure, of course, ok, okay, right, alright, no problem, not at all, never mind, please do, speaking

Table D.2: The hand-defined grammar for dialog act identification.

Appendix E

Category Inheritance Rules

The examples for the task goals COMPLAINT, ORDER, RESERVATION and SERVING are described below:

- The category inheritance rule for the COMPLAINT queries is illustrated in Table E.1 with a dialog example. In the example, the categories <Complain>, <MealDescription> and <Criticism> are inherited in the query "*All right, but please be quick.*". The categories inherit values from the latest customer dialog turn.

- The category inheritance rule for ORDER queries is illustrated in Table E.2 with a dialog example. In the example, the inheritance of both categories <MealDescription> and <Food_Drink> takes place on the query "*I'll have it medium-rare please.*", however the category <Food_Drink> referring to *'steak'* is not inherited in the next self-contained customer query "*What is today's special?*".

- The category inheritance rule for the RESERVATION queries is illustrated in Table E.3 with a dialog example. In the example, the cate-

gories <Arrange>, <Location>, <MealDescription>, <NumberValue>, <RelativeDate>, <RelativeTime>, <Reserve>, <SmokeOption>, <Table> and <Time> are inherited from the latest waiter or customer dialog turn.

- The category inheritance rules for the task goal SERVING is illustrated in Table E.4. For the SERVING queries (i.e. an inquiry about reqesting utensils or menu), the category <Utensil> should be inherited from previous waiter dialog turn if the current customer query contains the category <NumberValue> and the category <Food_Drink> should be inherited if the current query contains the category <Cook>.

Customer1:	*"I want to make a complaint for the meal!"* **TG Categories:** {<Complain = "complain"> <MealDescription = "meal">} **TG:** COMPLAINT
Customer1:	*"The steak is under-cooked!"* **TG Categories:** {<Food_Drink = "steak"> <Criticism = "under-cooked">} **TG:** COMPLAINT
Waiter1 :	*"I'm sorry, sir."* **TG Categories** {NIL } **TG:** COMPLAINT
Waiter1 :	*"I will return it to the chef immediately."* **TG Categories:** {<Chef = "chef"> } **TG:** COMPLAINT
Customer2:	*"Please be quick."* **TG Categories:** {NIL } **TG:** OOD
Customer2:	*"My steak should be well-done."* **TG Categories:** { <Food_Drink = "steak"> <SteakStyle = "well-done"> *<Complain = "complain">* *<MealDescription = "meal">* *<Criticism = "under-cooked">* } **TG:** COMPLAINT

Table E.1: Categories <Complain>, <MealDescription> and <Criticism> are selectively inherited for queries with task goal COMPLAINT. The categories in italics are inherited from discourse.

Customer1:	*"I'll have a sirloin steak for my meal."* **TG Categories:** {<Food_Drink = "sirloin steak"> <MealDescription = "meal"> } **TG:** ORDER
Waiter1 :	*"How would you like your steak, sir?"* **TG Categories** {<Food_Drink = "steak"> } **TG:** ORDER
Customer2:	*"I'll have it medium-rare please."* **TG Categories:** {<SteakStyle = "medium-rare"> *<Food_Drink = "steak"> <MealDescription = "meal">*} **TG:** ORDER
Waiter2 :	*"OK."* **TG Categories** {NIL } **TG:** OOD
Waiter2 :	*"Will there be anything else?"* **TG Categories** {<Else = "else"> } **TG:** ORDER
Customer3:	*"What is today's special?"* **TG Categories:** { <TodaySpecial = "today's special"> *<MealDescription = "meal">* } **TG:** ORDER

Table E.2: Categories <Food_Drink> and <MealDescription> are selectively inherited for queries with task goal ORDER. The categories in italics are inherited from discourse.

Customer1:	*"Can you arrange a dinner on the dining room for us?"* **TG Categories:** {<Arrange = "arrange">* <MealDescription = "dinner">* <Location = "dining room"> } **TG:** RESERVATION
Customer1:	*"I would like to reserve a table for four tommorrow evening."* **TG Categories:** {<Table = "table"> <Reserve = "reserve"> <NumberValue = "four"> <RelativeDate = "tommorrow"> <RelativeTime = "evening">} **TG:** RESERVATION
Waiter1 :	*"At what time can we expect you, sir?"* **TG Categories** {<What = "what"> <Time = "time"> } **TG:** RESERVATION
Customer2:	*"At seven."* **TG Categories:** { <Time = "seven"> *<Arrange = "arrange"> <MealDescription = "dinner">* *<Table = "table"> <NumberValue = "four">* *<RelativeDate = "tommorrow">* *<RelativeTime = "evening">* *<Reserve = "reserve"> <Location = "dining room">* } **TG:** RESERVATION
Waiter2 :	*"Would you like a table in the smoking area or non-smoking area, sir?"* **TG Categories** {<Table = "table"> <SmokeOption = "smoking"> <SmokeOption = "non-smoking"> } **TG:** RESERVATION
Customer3:	*"Non-smoking, please."* **TG Categories:** { <SmokeOption = "non-smoking"> *<Arrange = "arrange"> <MealDescription = "dinner">* *<Reserve = "reserve"> <Table = "table">* *<NumberValue = "four"> <RelativeDate = "tommorrow">* *<RelativeTime = "evening"> <Location = "dining room">* *<Time = "seven">* } **TG:** RESERVATION

Table E.3: Categories <Arrange>, <Location>, <MealDescription>, <Number -Value>, <RelativeDate>, <RelativeTime>, <Reserve>, <Smoke -Option>, <Table> and <Time> are selectively inherited for queries with task goal RESERVATION. The categories in italics are inherited from discourse.

Customer1:	*"Can you bring me some glasses?"* **TG Categories:** {<RelativeAmount = "some"> <Utensil = "glasses"> } **TG:** SERVING
Waiter1 :	*"How many glasses do you need, sir?"* **TG Categories** {<HowMany = "how many"> <Utensil = "glasses"> } **TG:** SERVING
Customer2:	*"Two, please."* **TG Categories:** { <NumberValue = "two"> *<Utensil = "glasses">* } **TG:** SERVING
Waiter2 :	*"Yes, sir"* **TG Categories** {NIL } **TG:** SERVING
Waiter2 :	*"I'll bring you immediately."* **TG Categories** {<Bring = "bring"> } **TG:** SERVING
Waiter2 :	*"Here's your steak, sir."* **TG Categories** {<Food_Drink = "steak"> } **TG:** SERVING
Customer3:	*"Can you help me to take it back and cook it longer?"* **TG Categories:** { <Take = "take"> <Cook = "cook"> *<Food_Drink = "steak">* } **TG:** SERVING

Table E.4: Categories <Utensil> and <Food_Drink> are selectively inherited for queries with task goal SERVING. The categories in italics are inherited from discourse.

Appendix F

Category Refresh Rules

The examples of the refresh rules are described below:

- Disinherit the category <Food_Drink> for ORDER queries if the customer rejects (i.e. the dialog act of the current query is identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previos waiter response is SUGGEST). , like the example dialog shown in Table F.1.

- Disinherit the category <Location> for RESERVATION queries if the customer rejects (i.e. the dialog act of the current query is identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previous waiter response is SUGGEST). Table F.2 illustrates this refresh rule with an example dialog .

- Disinherit the category <SmokeOption> for RESERVATION queries if the customer rejects (i.e. the dialog act of the current query is

identical to NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK) the suggestion from the waiter (i.e. the dialog act of previous waiter response is SUGGEST). Table F.3 illustrates this refresh rule with an example dialog .

- Disinherit the category <PriceValue> for BILL queries if the customer clarifies the bill value (i.e. the dialog act of the previous query is identical to REQUEST_INFO) but the waiter does not explicitly agree with the proposed bill value (i.e. the dialog act of waiter response is NEGATIVE_FEEDBACK or it is not identical to POSITIVE_FEEDBACK). Table F.4 illustrates this refresh rule with an example dialog.

Waiter1 :	*"Wanna try some desserts today?"* **Categories:** { <Try = "try"> <RelativeDate = "today"> <Food_Drink = "desserts"> <Quest = "?"> } **TG:** ORDER (from annotation) **DA:** SUGGEST (from annotation)
Customer2:	*"I prefer something else."* **Categories:** { <Perference_Phrase = "prefer"> <Else = "something else"> <Period = "."> ***<Food_Drink = "desserts"> (Over-inheritance)*** } **TG:** ORDER **DA:** NEGATIVE_FEEDBACK

Table F.1: An example dialog showing over-inheritance of the category <Food_Drink> for queries with task goal ORDER. This over-inheritance causes the system to misunderstand the customer to prefer the suggested "desserts".

Customer1:	*"I'd like to have a table for two in the main restaurant."* **Categories:** { <Perference_Phrase = "I'd like"> <Table = "table"> <NumberValue = "two"> <Period = "."> <Location = "main restaurant"> } **TG:** RESERVATION **DA:** PREFER
Waiter1 :	*"I'm afraid all our tables are filled up right now, sir. If you are in a hurry, we also serve dinner at coffee shop."* **Categories:** { <Table = "tables"> <Full = "filled up"> <RelativeTime = "right now"> <Period = "."> <Serve = "serve"> <MealDescription = "dinner"> <Location = "coffee shop"> <Period = "."> } **TG:** RESERVATION (from annotation) **DA:** INFORM, SUGGEST (from annotation)
Customer2:	*"No, thanks. I'll wait then."* **Categories:** { <No_Phrase = "no"> <Thank_Phrase = "thanks"> <Period = "."> <Wait = "wait"> <Period = "."> *<Table = "table">* *<NumberValue = "two"> <RelativeTime = "right now">* *<MealDescription = "dinner">* ***<Location = "coffee shop"> (Over-inheritance)***} **TG:** RESERVATION **DA:** NEGATIVE_FEEDBACK, THANK, INFORM

Table F.2: Over-inheritance of the category <Location> for queries with task goal RESERVATION. This over-inheritance causes the system to misunderstand the customer to prefer the suggested "coffee-shop".

Customer1:	*"A table for two in the non-smoking section, please."* **Categories:** { <Table = "table"> <NumberValue = "two"> <SmokeOption = "non-smoking"> <Please = "please"> <Period = "."> } **TG:** RESERVATION **DA:** INFORM
Waiter1 :	*"Sorry, sir. Would you rather take a table in the smoking section?"* **Categories:** { <Apology_Phrase = "sorry"> <Period = "."> <Take = "take"> <Table = "table"> <SmokeOption = "smoking"> <Quest_Mark = "?"> **TG:** RESERVATION (from annotation) **DA:** APOLOGY, SUGGEST (from annotation)
Customer2:	*"No, thanks. We'd like to wait then."* **Categories:** { <No_Phrase = "no"> <Thank_Phrase = "thanks"> <Period = "."> <Wait = "wait"> <Period = "."> *<Table = "table"> <NumberValue = "two">* ***<SmokeOption = "smoking"> (Over-inheritance)***} **TG:** RESERVATION **DA:** NEGATIVE_FEEDBACK, THANK, INFORM

Table F.3: Over-inheritance of the category <SmokeOption> for queries with task goal RESERVATION. This over-inheritance causes the system to misunderstand the customer to prefer the suggested "smoking" table.

Customer1:	*"How much is it?"* **Categories:** { <HowMuch = "how much"> <Quest_Mark = "?"> } **TG:** BILL **DA:** REQUEST_INFO
Waiter1 :	*"Your bill comes to $490."* **Categories:** { <Bill= "bill"> <PriceValue = "$490"> **TG:** BILL (from annotation) **DA:** INFORM (from annotation)
Customer2:	*"Oh. Shouldn't it be $450?"* **Categories:** { <Backchannel_Phrase = "oh"> <Request_Phrase = "shouldn't it"> <PriceValue = "$450"> <Quest_Mark = "?"> <*Bill = "bill"*> } **TG:** BILL **DA:** BACKCHANNEL, REQUEST_INFO
Waiter2 :	*"Your bill includes a ten percent service charge, sir."* **Categories:** { <Bill = "bill"> <PercentValue = "ten percent"> <ExtraCharge = "service charge"> <Period = "."> } **TG:** BILL (from annotation) **DA:** INFORM (from annotation)
Customer3:	*"OK. Do you accept credit card?"* **Categories:** { <Yes_Phrase = "ok"> <Accept = "accept"> <PayMethod = "credit card"> <*Bill = "bill"*> ***<PriceValue = "$450"> (Over-inheritance)*** } **TG:** BILL **DA:** POSITIVE_FEEDBACK, REQUEST_INFO

Table F.4: Over-inheritance of the category <PriceValue> for queries with task goal BILL. This over-inheritance causes the system to misunderstand the bill to be "$450".

Appendix G

Full list of Response Trigger Words

Response Trigger Word	Text Responses
ano_advise_reserve	We would advise you to make a reservation.
ano_call_me	You can call me whenever you need help.
ano_confirm_asking_equipment	#Equipment .
ano_confirm_asking_meal_time	Time for #MealDescription .
ano_confirm_asking_what_food_serve	The food.
ano_confirm_closing_time	Closing time.
ano_confirm_food_call	The Name of this food .
ano_confirm_opening_time	Opening time.
ano_confirm_price_range	The price range.
ano_confirm_reserve	Reservation.
ano_inform_closing_time	Our restaurant closes at #Time .
ano_inform_equipment_loc	It's #Location .
ano_inform_food_call	It is called #Food_Item .
ano_inform_meal_time	Our restaurant is opened for breakfast from #BreakfastTime until #LunchTime , and dinner from #DinnerTime .
ano_inform_opening_time	Our restaurant opens at #Time .
ano_inform_price_range	It depends. Around fifty to two hundred dollars per person.

ano_inform_what_food_serve	We serve a great variety of popular american dishes.
ano_req_asking_equipment	#Equipment ?
ano_req_asking_meal_time	Time for #MealDescription ?
ano_req_asking_what_food_serve	The food?
ano_req_closing_time	Closing time?
ano_req_food_call	The name ?
ano_req_opening_time	Opening time?
ano_req_price_range	The price range?
ano_req_reserve	Reservation?
any_else	Anything else I can help you?
any_else_offer_help	Anything else I can help you, sir?
apology	We are sorry, sir./We are very sorry, sir.
backchannel	I see./hmm.../well.../huh...
bil_accept_pay_method	We can accept #PayMethod here.
bil_cash	Can you pay it in cash?
bil_explain_bill	It includes the tax and service charges, sir.
bil_inform_bill	Your bill goes to *#PriceValue./Youbillwillbe* #PriceValue ./Your bill goes to to *#PriceValue, sir./Youbillwillbe* #PriceValue , sir.
bil_inform_changes	Here is your changes.
bil_pay_me	Pay me, please.
bil_pay_method_worth	It's worth #PayMethodWorth .
bil_reject_pay_method	I'm afraid we cannot accept #PayMethod here.
bye	Bye./Bye bye./Goodbye.
call_me	You can call me whenever you need help.
close	You are welcome.
com_bring_one	I will bring you one.
com_change	I'll change it for you.
com_check_order	I will check your order first.
com_commit_new_one	I will bring you a new one.
com_right_dish	I'll bring you the right dish at once.
commit	Yes I will.
defer	Please wait a second./Wait a moment, please.
greeting	Hello./Hi.
have_a_nice_day	Have a nice day.
hope_enjoy_meal	I hope you enjoy your meal.
introduce	My name is Harry. I will be your server.

no	No./No, sir.
no_music	We do not provide music here.
offer_help	Can I help you, sir?/Can I help you?/What can I help you?/May I help you?
offer_serve	May I serve the food to you now?
offer_what_can_i_help	What can I help you?/What can I help you, sir?
ood_inform	We do not provide such service.
ord_anything_else	Anything else, sir?/Is that all?/Is that anything else?/Is there anything else, sir?
ord_ask_egg_style	How would you like us to cook your eggs? sunny side up, over easy, hard boiled, poached or scrambled?
ord_ask_ham_or_bacon	We serve ham or bacon with your eggs. Which would you prefer?
ord_ask_juice	What kind of juice would you prefer, tomato or orange or apple?
ord_ask_now_or_later	Would you like it now or later?
ord_ask_roll_or_toast	Would you like it with toasts or rolls?
ord_ask_steak_style	How would you like that done? rare, medium or well done?
ord_ask_tea_or_coffee	And tea or coffee?/Would you like tea or coffee?
ord_ask_what_to_eat	What would you like to eat?
ord_ask_whether_fine	Will this be fine?
ord_ask_whether_good	Is the food fine?
ord_charge_extra	we charge you extra for it.
ord_come_food	It is come with #Food_Item .
ord_commit	I'll make sure the chef prepares it just the way you like.
ord_commit_food	OK. #Food_Item.
ord_commit_time	Your order should arrive within 15 minutes.
ord_confirm_food	You have ordered #Food_Item .
ord_inform_dressing	It's made of #Ingredient .
ord_inform_food	#Food_Item .
ord_inform_size	We serve #Size , sir.
ord_inform_sub	The only thing I can substitute for you is #Food_Item .
ord_inform_time_arrive	It should take about fifteen minutes, sir.
ord_inform_today_special	We would have #Food_Item for today's special.
ord_inform_unit_food	OK. I will give #Unit of #Food_Item to you later.

ord_it's_desc	It's #Description .
ord_it's_good	It's very well, sir./It's good, sir./It's very good, sir./It's really good./It's really good, sir.
ord_offer	May I take your order, sir?/What would you like to order, sir?
ord_provide_food	These are the food we provided for you.
ord_serve_by_equip	You serve it by #Equipment .
ord_suggest	How about #Food_Item ?/What about #Food_Item ?/I would recommend #Food_Item .
ord_whether_decide_sth	Have you decided on something?
ord_whether_try	Wanna try some?/Do you want to try it, sir?
ord_you_can_pay_me	You can pay me in cash for your meal.
req_comment	Is it OK?/Is it alright?/Is everything alright?/Is everything OK?
res_ask_location	Where would you like to sit? By the window , in the main restaurant or in the bar?
res_ask_name_phone	May I have your surname and phone number, please?
res_ask_peope_num	For how many persons, please?
res_ask_smoke_option	Would you like to have a seat in smoking area or non-smoking area?
res_ask_time	At what time can we expect you?
res_ask_whether_reserve	Do you have any reservation?
res_commit_seat_soon	We can seat you very soon.
res_commit_show_table	I'll show your table.
res_confirm_name	#Name .
res_confirm_reservation	You have reserve a table for #NumberValue , #Location at #Time #RelativeDate .
res_inform_table_available	We have a table for you now.
res_inform_table_full	I'm afraid that we are full right now.
res_is_table_fine	Will this table be fine?
res_please_seat	Please take a seat, sir.
res_show_new_table	I 'll show you a new table.
res_suggest_other_location	Would you like having a table in #Location ?
res_suggest_other_smokeoption	Would you like having a table in #SmokeOption ?
res_suggest_wait	Would you wait for a while?
res_table_ready	Your table is ready./We are expecting you.

res_this_way	This way, please.
res_wait_apology	We are very sorry to have kept you waiting.
res_which_day	Which day do we expect?
see_you	See you then.
ser_call_me	You can call me whenever you need help.
ser_enjoy_meal	Please enjoy your meal.
ser_menu_here	Here is the menu.
ser_move_plate	May I move your plate to the side?
ser_serve_food	Your #Food_Item, sir.
ser_take_time	Please take your time.
thank	Thank you, sir.
waiter_will_come_to_order	A waiter will come and take your order.
welcome_phrase	Welcome to HCI restaurant.
what	Pardon me, sir. What can I help you?
yes	Yes, sir./Certainly, sir.

Table G.1: Full list of response trigger words.

Appendix H

Evaluation Test Questionnaire for Dialog System in CU Restaurants Domain

Evaluation of Dialog Management in the CU Restaurants Domain

Background.

In the restaurant, there is a virtual waiter who can serve you anything in the restaurants domain, such as restaurant reservations, food ordering, billing, complaints and information answering. You are now trying to communicate with him in order to complete the following tasks. After finishing each task, please answer the questions.

Task One. You want to reserve a table with the following details:

i. Time of Reservation: 7:30 pm tomorrow.
ii. Number of people: five
iii. Location of the table you want: by the window

Questions: (Rates from 1 to 5: 1="very bad", 5 = "very good")

1. Do you think the answers of the virtual waiter are relevant to your questions? ( )
2. Do you think the answers of the virtual waiter are clear? ( )
3. Do you think the virtual waiter has made true statements? ( )
4. Do you think the answers of the virtual waiter are informative as you expected? ( )
5. Do you satisfy with the performance of the dialog manager for this task? ( )

Part Two. You are now trying to order:

i. One smoked turkey with green salad, and
ii. A cup of Chinese tea

Questions: (Rates from 1 to 5: 1="very bad", 5 = "very good")

6. Do you think the answers of the virtual waiter are relevant to your questions? ( )
7. Do you think the answers of the virtual waiter are clear? ( )
8. Do you think the virtual waiter has made true statements? ( )
9. Do you think the answers of the virtual waiter are informative as you expected? ( )
10. Do you satisfy with the performance of the dialog manager for this task? ( )

Part Three. You want to have a bill now.

Questions: (Rates from 1 to 5: 1="very bad", 5 = "very good")

11. Do you think the answers of the virtual waiter are relevant to your questions? ( )
12. Do you think the answers of the virtual waiter are clear? ( )
13. Do you think the virtual waiter has made true statements? ( )
14. Do you think the answers of the virtual waiter are informative as you expected? ( )
15. Do you satisfy with the performance of the dialog manager for this task? ( )

Figure H.1: The evaluation test questionnaire to evaluate the user satisfaction of our dialog system in the CU Restaurants domain.

Appendix I

Details of the statistical testing Regarding Grice's Maxims and User Satisfaction

The parameter of interest is the difference in **Maxim of Relevance** score μ_I of the task RESERVATION
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.35, s = 0.81, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.35-3}{0.81/\sqrt{20}} = 1.926$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.926 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Relevance for the task RESERVATION

Figure I.1: Details of statistical testing on the task RESERVATION of our dialog system regarding Maxim of Relevance.

The parameter of interest is the difference in **Maxim of Manner** score μ_I of the task RESERVATION
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.45, s = 0.83, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.45-3}{0.83/\sqrt{20}} = 2.44$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 2.44 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Manner for the task RESERVATION

Figure I.2: Details of statistical testing on the task RESERVATION of our dialog system regarding Maxim of Manner.

The parameter of interest is the difference in **Maxim of Quality** score μ_I for the task RESERVATION
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.45, s = 1.05, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.45-3}{1.05/\sqrt{20}} = 1.92$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.92 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quality for the task RESERVATION

Figure I.3: Details of statistical testing on the task RESERVATION of our dialog system regarding Maxim of Quality.

The parameter of interest is the difference in **Maxim of Quantity** score μ_I for the task RESERVATION
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.55, s = 0.83, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.55-3}{0.83/\sqrt{20}} = 2.98$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 2.98 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quantity for the task RESERVATION

Figure I.4: Details of statistical testing on the task RESERVATION of our dialog system regarding Maxim of Quantity.

The parameter of interest is the difference in **User Satisfaction** score μ_I for the task RESERVATION
$H_0 : \mu_I = 0$
$H_1 : \mu_I > 0$
$\alpha = 0.05$ The test statistic is
$t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.35, s = 0.81, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.35-3}{0.81/\sqrt{20}} = 1.93$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.93 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of User Satisfaction for the task RESERVATION

Figure I.5: Details of statistical testing on the task RESERVATION of our dialog system, regarding Perceived User Satisfaction.

The parameter of interest is the difference in **Maxim of Relevance** score μ_I of the task ORDER
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.9, s = 0.55, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.9-3}{0.55/\sqrt{20}} = 7.28$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 7.28 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Relevance for the task ORDER

Figure I.6: Details of statistical testing on the task ORDER of our dialog system regarding Maxim of Relevance.

The parameter of interest is the difference in **Maxim of Manner** score μ_I of the task ORDER
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.8, s = 0.77, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.8-3}{0.77/\sqrt{20}} = 4.66$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 4.66 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Manner for the task ORDER

Figure I.7: Details of statistical testing on the task ORDER of our dialog system regarding Maxim of Manner.

The parameter of interest is the difference in **Maxim of Quality** score μ_I for the task ORDER
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.9, s = 0.85, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.9-3}{0.85/\sqrt{20}} = 4.72$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 4.72 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quality for the task ORDER

Figure I.8: Details of statistical testing on the task ORDER of our dialog system regarding Maxim of Quality.

The parameter of interest is the difference in **Maxim of Quantity** score μ_I for the task ORDER
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.9, s = 0.72, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.9-3}{0.72/\sqrt{20}} = 5.6$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 5.6 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quantity for the task ORDER

Figure I.9: Details of statistical testing on the task ORDER of our dialog system regarding Maxim of Quantity.

The parameter of interest is the difference in **User Satisfaction** score μ_I for the task ORDER
$H_0 : \mu_I = 0$
$H_1 : \mu_I > 0$
$\alpha = 0.05$ The test statistic is
$t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.8, s = 0.89, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.8-3}{0.89/\sqrt{20}} = 4$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 4 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of User Satisfaction for the task ORDER

Figure I.10: Details of statistical testing on the task ORDER of our dialog system, regarding Perceived User Satisfaction.

The parameter of interest is the difference in **Maxim of Relevance** score μ_I of the task BILL
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.5, s = 1.19, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.5-3}{1.19/\sqrt{20}} = 1.88$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.88 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Relevance for the task BILL

Figure I.11: Details of statistical testing on the task BILL of our dialog system regarding Maxim of Relevance.

The parameter of interest is the difference in **Maxim of Manner** score μ_I of the task BILL
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.55, s = 1.23, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.55-3}{1.23/\sqrt{20}} = 1.99$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.99 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Manner for the task BILL

Figure I.12: Details of statistical testing on the task BILL of our dialog system regarding Maxim of Manner.

The parameter of interest is the difference in **Maxim of Quality** score μ_I for the task BILL
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.5, s = 1.19, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.5-3}{1.19/\sqrt{20}} = 1.88$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.88 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quality for the task BILL

Figure I.13: Details of statistical testing on the task BILL of our dialog system regarding Maxim of Quality.

The parameter of interest is the difference in **Maxim of Quantity** score μ_I for the task BILL
$H_0 : \mu_I = 3$
$H_1 : \mu_I > 3$
$\alpha = 0.05$
The test statistic is $t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.55, s = 1.23, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.55-3}{1.23/\sqrt{20}} = 1.99$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Conclusion: Since $t_0 = 1.99 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of Maxim of Quantity for the task BILL

Figure I.14: Details of statistical testing on the task BILL of our dialog system regarding Maxim of Quantity.

The parameter of interest is the difference in **User Satisfaction** score μ_I for the task BILL
$H_0 : \mu_I = 0$
$H_1 : \mu_I > 0$
$\alpha = 0.05$ The test statistic is
$t_0 = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$
Reject H_0 if $t_0 > t_{00.5,20} = 1.725$
Computation: $\bar{x} = 3.5, s = 1.19, \mu_{IO} = 3, n = 20$, we have
$t_0 = \frac{3.5-3}{1.19/\sqrt{20}} = 1.88$
Reject H_0 if $t_0 > t_{00.5,20} = 1.88$
Conclusion: Since $t_0 = 1.88 > 1.725$, we reject H_0 and conclude at the 0.05 level of significance that our system achieves better results than average in terms of User Satisfaction for the task BILL

Figure I.15: Details of statistical testing on the task BILL of our dialog system, regarding Perceived User Satisfaction.