Learning the Structure of Task-Oriented Conversations from the Corpus of In-Domain Dialogs

Ananlada Chotimongkol

CMU-LTI-08-001

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Alexander I. Rudnicky, Chair
William W. Cohen
Carolyn Penstein Rosé
Gokhan Tür, SRI International

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies*

© 2008, Ananlada Chotimongkol

Keywords: domain knowledge acquisition, dialog structure model, task-oriented dialogs, spoken dialog system, dialog structure annotation, unsupervised learning, word clustering, document clustering, text segmentation

Acknowledgements

I would like to take this opportunity to thank all the people who have helped me with my research and thus made this work possible. First of all, I am very grateful to my advisor, Alexander Rudnicky, for his valuable guidance and support throughout the course of this work and my graduate school career. His vision and understanding in scientific research teach me all the elements I would need to know when conducting high quality research on my own. I am also sincerely grateful to my committee members, William Cohen, Carolyn Penstein Rosé, and Gokhan Tür, for their valuable suggestions and contributions to this research. Carolyn Penstein Rosé in particular gave me a great deal of feedback on the thesis writing.

While at CMU, I have been lucky enough to be a part of a friendly and very helpful research community. I would like to thank my fellow research group members and the members of the Dialogs on Dialogs reading group, Dan Bohus, Antoine Raux, Mihai Rotaru, Banerjee Satanjeev, Stefanie Tomko, and many other people that I would not be possible to list all their names here, for numerous intellectual discussions and their encouragement. I also would like to thank Rose Hoberman for the mutual information-based clustering program, Jaime Arguello Museli for the HMM-based segmentation program, and Banerjee Satanjeev for the dialog segmentation program and evaluation.

I have been fortunate to have great friends at CMU, whose continuous support was invaluable. And above all their friendship has made the life in Pittsburgh a home far away from home. In particular, I would like to thank my roommates, Nawaporn Wisitpongphan and Samita Dhanasobhon, and also Win Trivitayanurak and Teeraphong Mahatham for being there through good and bad times. I also would like to express my gratefulness to Pahnit Seriburi, my long-time friend, who always being there for me.

My gratitude also goes to my former advisor Surapant Meknavin who led me into the field of natural language processing and who has continued to give me advice throughout the years. And the last but not least, I would like to thank my parents for their love and encouragement, and for allowing their only child to be very far away from home to pursuit her dream in academic achievement.

Table of Contents

List of Figures

List of Tables

Abstract

Acquiring domain-specific knowledge necessary for creating a dialog system in a new task-oriented domain is a time consuming task that requires domain expertise. This dissertation explores the feasibility of using a machine learning approach to infer the required domain-specific information automatically from in-domain conversations. In order to achieve this goal, two problems need to be addressed: 1) creating a dialog representation that is suitable for representing the required domain-specific information, and 2) developing a machine learning approach that uses this representation to capture information from a corpus of in-domain conversations.

In order to solve the first problem, I propose a form-based dialog structure representation incorporating a three-level structure of *task*, sub-*task*, and *concept*. These components are observable in human dialogs. In terms of representation, tasks and sub-tasks are represented by forms while concepts are slots in a form. The notion of *form* is generalized as a repository of related pieces of information so that it can be applied to various types of task-oriented domains. Dialog structure analysis and an annotation experiment are used to demonstrate that the form-based representation has all the required properties: *sufficiency*, *generality*, and *learnability*. The proposed representation is applied to six disparate task-oriented domains (air travel planning, bus schedule inquiry, map reading, UAV flight simulation, meeting, and tutoring). While the form-based approach shows some limitations, it is sufficient to model important phenomena in dissimilar types of task-oriented dialogs, and thus has both *sufficiency* and *generality*. The annotation experiment shows that the form-based dialog structure representation can be applied reliably by novice annotators which implies that the representation is unambiguous and *learnable*.

For the second problem, inferring the form-based dialog structure representation from a corpus of in-domain conversations, I divide this dialog structure acquisition problem into two sub-problems, concept identification and form identification, to make the problem tractable. In order to identify a set of domain concepts, two unsupervised concept clustering approaches are investigated: statistical-based clustering and knowledge-based clustering. For most statistical-based clustering algorithms, we are able to find automatic stopping criteria that yield close to optimal results. The statistical-based approaches, which utilize word co-occurrence statistics such as mutual information and the Kullback-Liebler distance, while able to capture domain-specific usage of concept words cannot accurately identify infrequent concept words due to a sparse data problem. On the other hand, the knowledge-based approach, which uses semantic information stored in the WordNet lexical database, can identify domain concepts very accurately, but on the condition that the concepts are present in the knowledge source.

To determine different types of forms and their associated slots, a dialog is first segmented into a sequence of sub-tasks by an unsupervised text segmentation algorithm. Then, the bisecting *K*-mean sub-task clustering algorithm is used to group the sub-tasks that represent the same form type into the same cluster. Finally, a set of slots that is associated with each form is determined from the concepts present in each cluster. To handle fine-grained segments in spoken dialogs, TextTiling and HMM-based segmentation algorithms are augmented with a data-driven stop word list and distance weights. With these modifications significant improvement is achieved. Even though the performance of the bisecting *K*-mean sub-task clustering algorithm can be affected by inaccurate sub-task boundaries, I found that moderate segmentation accuracy is sufficient for identifying frequent form types. Similarly, moderate sub-task clustering accuracy is sufficient for identifying essential slots in each form.

The results of both dialog structure acquisition problems, concept identification and form identification, show that it is feasible to acquire the domain-specific knowledge necessary for creating a task-oriented dialog system automatically from a corpus of in-domain conversations using unsupervised learning approaches. This data-driven approach could potentially reduce human effort in developing a new task-oriented dialog system.

Chapter 1

Introduction

A spoken dialog system is a computer system that interacts with a user via natural spoken language to help the user obtains desired information or resolves a problem. As for current technologies, a dialog system is one of many natural language applications that operate on a specific domain. For instance, the CMU Communicator system (Rudnicky et al., 1999) is a dialog system in an air travel domain that provides information about flight, car, and hotel reservations. Another example is the JUPITER system (Zue et al., 2000) which is a dialog system in a weather domain that provides forecast information for a requested city.

A dialog system typically is composed of the following components: a speech recognizer, a natural language understanding module, a dialog manager, a natural language generation module and a speech synthesizer. When developing a dialog system in a new domain, we may be able to reuse some components from existing dialog systems if they were designed independently of domain-specific information. Examples of such domain-independent components include speech recognizer and speech synthesizer engines. However, the components that are integrated with domain-specific information have to be modified or reconstructed for every new domain. In a task-oriented dialog, which is the type of dialog that is focused in this thesis, participants engage in a conversation in order to achieve a specific goal (i.e. to accomplish a task that they have in mind), for example, to obtain the departure time of a particular bus or to order a product from a catalog. Hence in the context of this thesis, domain-specific information refers to the knowledge which is specific to a task that a dialog system has to support rather than the knowledge about general dialogue mechanisms. The work in this thesis also focuses specifically on the domain-specific knowledge that is used by a dialog manager rather than the one that might be required by other components in the system. The domain-specific information used by a dialog manager includes a list of tasks that a dialog system has to support, a list of steps that needs to be taken in order to successfully accomplish each task, and domain keywords which capture pieces of information that dialog participants need to communicate in order to achieve these steps. An example of the necessary domain knowledge in an air travel domain is shown in Figure 1.2.

Conventionally when developing a dialog system in a new domain, the domain-specific information is identified manually by a human who is familiar with the domain. For common domains, such as an air travel domain or a weather domain, dialog system developers usually have enough knowledge to identify the necessary domain-specific information. However, for unfamiliar domains, such as a military domain (Bohus and Rudnicky, 2002), the necessary expertise may be scarce which makes the knowledge engineering process more difficult. Furthermore, a domain expert's decision can be subjective and may not cover all of the possible cases as users' perspectives of a task may not be foreseen by the expert (Yankelovich, 1997). As more dialog data becomes available, recent approaches in acquiring the domain knowledge are more data-driven. Dialog system developers identify the necessary domain information by analyzing conversations between the humans who perform similar tasks as a target dialog system. The use of in-domain data can be supplemented for the need of a domain expert. A data-driven approach is less subjective and also reflects more realistic users' perspectives of a task. However, the main drawback of this approach is that analyzing in-domain conversations manually is very time consuming.

In the past decade, the computational linguistics community has focused on developing language processing algorithms that can leverage the vast quantities of corpus data that are available. The same idea can also be applied to the problem of acquiring domain-specific information. A machine learning technique could potentially reduce human effort in the knowledge engineering process and alleviate the bottleneck in developing a new dialog system. This thesis investigates the possibility of using a machine learning approach to acquire the domain-specific information required to build a task-oriented dialog system from in-domain conversations. Figure 1.1 outlines the proposed solution to the problem of domain knowledge acquisition. Instead of identifying the required domain knowledge from in-domain dialogs manually, the knowledge engineering process will be done automatically using a machine learning approach. The automatically obtained information, such as the one shown in Figure 1.2, can be revised by dialog system developers before being used to build a dialog system. Even though some revision might be required, the amount of effort spent on revising the learned information should be smaller than the amount of effort spent on manually analyzing in-domain dialogs. This thesis focuses on the highlighted part, inferring the required domain knowledge from in-domain dialogs.

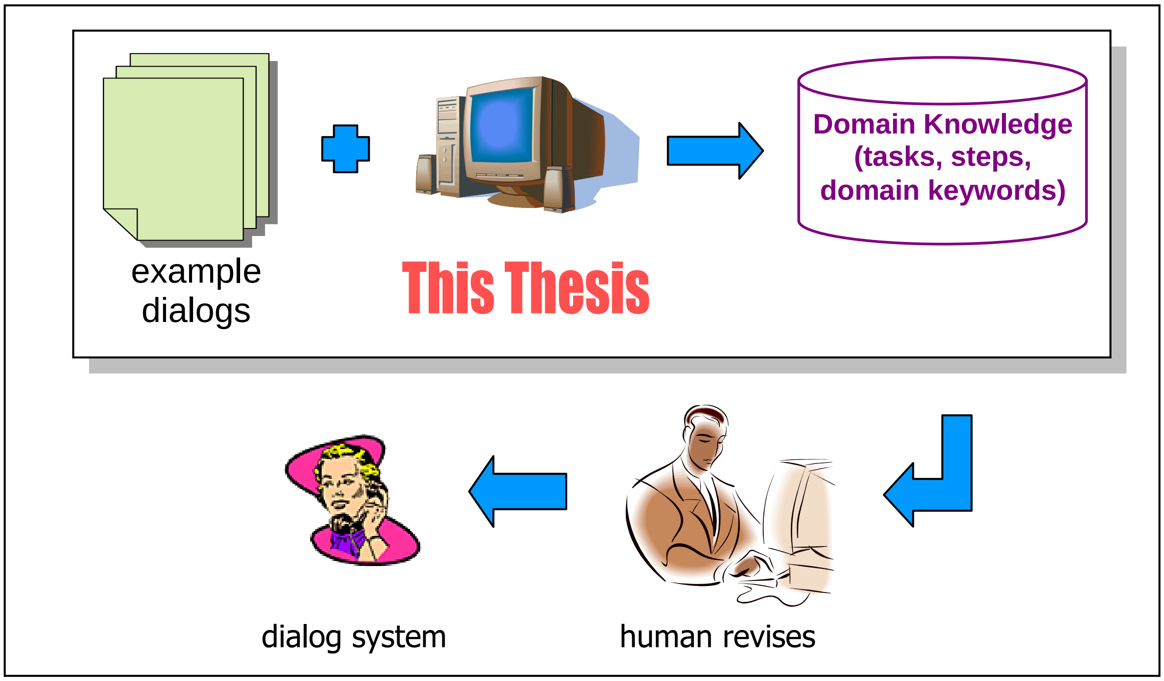


Figure 1.1: The proposed domain knowledge acquisition process

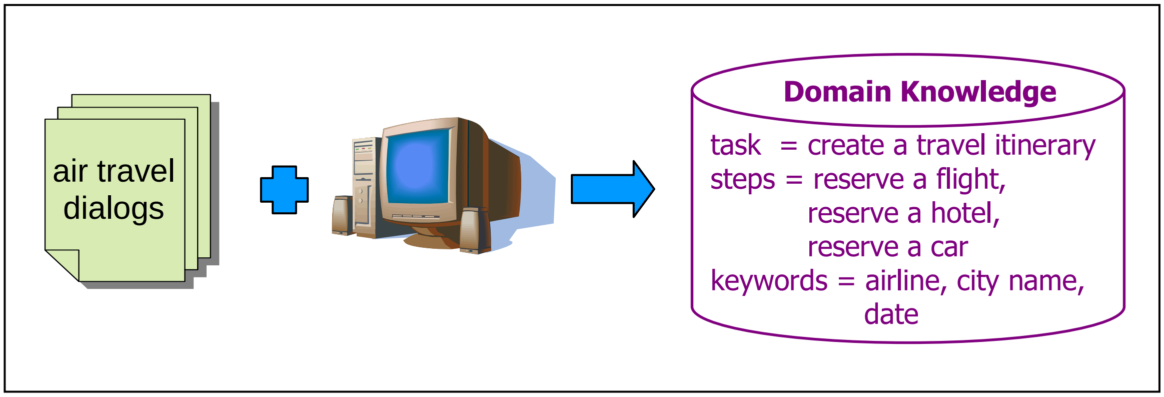

air travel
dialogs
Domain Knowledge
task = create a travel itinerary
steps = reserve a flight,
reserve a hotel,
reserve a car
keywords = airline, city name,
date

Figure 1.2: An example output of the proposed machine learning approach

In the following sections, I will provide some background knowledge about task-oriented dialog systems and then outline the proposed solution to the problem of acquiring the domain-specific information necessary for creating a task-oriented dialog system. The rest of this chapter is organized as follows: Section 1.1 provides background knowledge on different types of dialog system architectures. Section 1.2 discusses the observable structure of a task-oriented dialog and how it reflects the domain-specific information required to build a dialog system. Section 1.3 discusses conventional approaches that have been used to develop a dialog system in a task-oriented domain. Section 1.4 provides the overview of the proposed solution to the problem of domain knowledge acquisition. The thesis statement is given in Section 1.5. Finally, the outline of the remainder of this thesis is provided in Section 1.6.

1.1 Dialog system architecture

The architecture of a dialog system governs the interaction between the system and a user as it sets up expectation on user input, guides the system actions and controls the flow of a dialog. Furthermore, a dialog itself and other related information, such as domain and world knowledge, are modeled differently by each type of dialog architecture. According to McTear (2002), dialog systems can be classified into three main categories based on the architecture of the systems: *a finite state-based system*, *a form-based system* and *an agent-based system*.

1.1.1 Finite state-based systems

In a finite state-based system (or *a graph-based system*), a dialog is modeled as a sequence of steps or *states*. The system takes an initiative and leads a user through a dialog graph or a state transition network. At each state, the system produces a specific prompt to elicit particular responses from the user. Based on the response, the system performs an action and transits to a new state. A set of dialog states and eligible transitions among the states are fixed and pre-determined. Domain knowledge and other information related to a dialog are modeled implicitly by dialog states and their transitions. Examples of finite state-based systems are the automatic book club service of Larsen and Baekgaard (1994) and the bus schedule information system of Bennett et al. (2002).

A major advantage of the finite state model is its simplicity. Since the system takes control over the interaction and user's responses at each dialog state are quite constrained, technologies required to build each system component are less demanding. Restricted input and simple interaction lead to fewer errors as discussed in Section 5.1.2 of McTear's article (2002). For those reasons, the finite state-based dialog architecture has been adopted in many commercial systems. Nevertheless, its simplicity can become a drawback as the model is not flexible enough to handle any deviation from expected interaction. The state-based dialog architecture is suitable for a well-structured task that has a clearly defined set of information items a dialog system needs to obtain and that the order for eliciting those items can be fixed. However, it is not appropriate for a complex task whose dialog flow cannot be pre-determined, (e.g. a task that requires negotiation or contains unknown constraints) or can be affected by the dependencies between information items.

1.1.2 Form-based systems

A form-based system (or *a frame-based* or *a template-based system*) models a dialog as an information gathering session analogous to a form-filling task. At each turn, the system attempts to elicit a piece of necessary information from a user and uses that information to fill the corresponding slot in a *form*. In the form-based dialog system architecture, a form is a repository of items of information required to perform a task such as acquiring flight information from a database. However, other data structures can also be used to represent those items of information as well. Examples of other data structures are a *product*, a collection of forms (Constantinides et al., 1998) and a *typed feature structure* (Denecke and Waibel, 1997). The Philips train timetable information system (Aust et al., 1995) and the CMU Communicator system (Rudnicky et al., 1999) are examples of real world applications that use the form-based dialog system architecture.

The information gathering process in the form-based system is quite similar to the one in the state-based system; however, the form-based architecture allows more flexible and more natural interaction. The flow of a dialog is determined dynamically from dialog context rather than pre-defined. An appropriate system action (e.g. which question the system should ask next) is chosen by a control algorithm or a dialog strategy based on dialog context such as the current content of a form and the user's previous utterance. The form-based system can handles more open input from a user; nevertheless, the understanding process focuses primarily on words or phrases that can be filled into a pre-defined set of lots. The user can take initiative by providing more information in addition to the one requested by the system. More flexible interaction makes a dialog more efficient but at the same time requires a more complicated control strategy. In the form-based system, domain-specific information is modeled explicitly by forms and slots and is decoupled from a control mechanism.

Similar to the state-based system, the form-based system is appropriate for a well-defined task. Even though a dialog flow does not have to be pre-determined as in the state-based system, a set of information items the form-based system has to elicit need to be specified. The system also utilizes only simple context information to determine appropriate actions. Therefore, it is not suitable for a domain that has a dynamic structure or requires complex interpretation of dialog context beyond the information represented by a form and dialog history. Moreover, since the form-based architecture assumes that a dialog is an information gathering interaction where the system acquires necessary information to perform a task from the user, this type of system cannot handle the dialogs that deviate from this assumption.

1.1.3 Agent-based systems

An agent-based system models a dialog as collaboration between intelligent agents and utilizes Artificial Intelligence (AI) techniques to manage the interaction between the system and a user. There are many variants of agent-based systems depending on intelligent behaviors and discourse theories adopted by the systems. For example, a theorem proving approach was used by Smith and Hipp (1994) while a plan-base approach and a rational interaction approach were used by Sadek et al.(1997). Generally, an agent-based system follows a human reasoning process by taking into account its own goals, beliefs and intentions and sometimes those of the user when determining an appropriate action. AI approaches mentioned above provide theoretical foundation on how those conceptual components influent the interaction between the system and the user as they collaborate in order to accomplish a task. Since the conceptual components (e.g. beliefs and intention) sometimes are not explicitly expressed in a dialog, a sophisticated natural language processing technique together with the knowledge about human conversations and a domain are required in order to infer those components from dialog context.

With a complex dialog model, an agent-based system can support a complicated task that the flow of a dialog evolves dynamically and the content of the interaction (e.g. the topics and key information discussed) cannot be determined in advance. For instance, during a tutoring session additional topics may need to be discussed if a tutoring system discovers that a student lacks the basic knowledge required to solve the current problem. The user can also take initiative by introducing a new topic or shifting from the current topic to a different one. Since both the sequence and the content of user input are not pre-defined, an understanding module needs to be able to handle fairly unrestricted user input. Examples of agent-based systems are the TRAINS system, a dialog system that helps a manager solve a routing problem in a transportation domain (Allen et al., 1995) and the physics tutoring system (Freedman, 2000).

The agent-based dialog system architecture can support a more complex task than the finite state-based and the form-based architectures. However, a more complicated model comes with the cost of intensive computation both in terms of dialog control and user's input interpretation.

1.1.4 Dialog system architecture comparison

The agent-based dialog system architecture seems to be more appealing than other types of architectures as it can model a more complex dialog and has the closest conversational competent to a human participant. However, for a simple and well-structured task, the finite state-based and the form-based architectures can be more efficient as they take less effort to develop. Simpler interaction is also more robust to system errors such as speech recognition errors and understanding problems. Table 1.1 summarizes the comparison among the three dialog system architectures along three aspects: user input, dialog flow and domain information.

Features	**Dialog system architecture**		
	State-based	**Frame-based**	**Agent-based**
User input	Single words or phrases restricted by system prompts	Natural language with concept spotting	Unrestricted natural language
Dialog flow	Fixed and pre-determined by a state transition diagram	Determined dynamically from the current content of a form and the user's previous utterance	Determined dynamically from a model of goals, beliefs and intentions
Domain information	Represented implicitly in terms of dialog states	Represented explicitly by forms and their associated slots	Represented explicitly in a knowledgebase and a domain reasoning module

Table 1.1: Dialog system architecture comparison

1.2 Characteristic of task-oriented conversations

Observations of goal-oriented human-human dialogs from different domains show that such dialogs have clear structures that capture domain-specific information. When two or more people engage in a conversation that has a specific goal, such as obtaining bus schedule information, they organize their conversion so that key ideas are clearly communicated and that progress towards the goal is observable by all the parties. If the task the dialog participants try to achieve is complicated, they usually divide the task into a series of sub-tasks in which they will pursue one at a time. This observation is similar to Grosz's (1978) discussion about dialog structure and task structure. A sub-task is accomplished through a domain action and all pieces of information required in order to perform the action have to be clearly communicated among the participants. The characteristics of task-oriented conversations are reflected in the choice of language, which will be instrumental, and will reference the shared representation of a task. This contrasts with the characteristics of non-task-oriented conversations such as the ones in the Switchboard corpus (Godfrey et al., 1992) and the CALLHOME corpus (Kingsbury et al., 1997). Those corpora are casual social conversations.

1.3 Conventional approaches in dialog system development

The development process of a dialog system is similar to development processes of other types of computer systems. This process involves specifying system requirements, designing and implementing a dialog system that meets all of the requirements, and evaluating the implemented system. Establishing system requirements is the first step in the process. In this step, dialog system developers need to specify the scope of a target dialog system (i.e. the tasks that the system can support and the functionality of the system); determine the structure of each task; indicate the desired interaction between the system and a user which includes determining the dialog flow and anticipated user input and system output; and list other technical constraints such as a user pool (native vs. non-native) and usage environment. This section focuses on this step, requirement specification, as it involves identifying domain-specific information.

In the current dialog system development process, dialog system requirements are specified manually by dialog system developers based on the knowledge that they have about the domain. However, the resulting system may not interact with a user in the way that he/she expects it to, as the developers' perspectives of the system could be different from the user' perspectives. To avoid this problem, the analysis of in-domain conversations is used to guide the design decisions made by the dialog system developers. For example, the dialog processing component in the VERBMOBIL speech-to-speech translation system was designed based on the analysis of scheduling dialogs and the requirements of other components in the system (Alexandersson and Reithinger, 1995). There are two ways to obtain a collection of in-domain conversations: 1) by recoding conversations between humans that perform the same task as a target dialog system, or 2) by simulating a target dialog system using a human wizard and recording conversations between a user and the simulated system.

The first method creates a corpus of human-human conversations which provides an insight into how human participants interact through a dialog to accomplish a task in a given domain. There are two types of human-human conversations the one that occurs in a real situation, e.g. a call made to a help desk operator of the Pittsburgh Port Authority Transit system (Raux et al., 2003), and the one that based on a prescribed scenario, e.g. the TRAINS corpus (Gross et al., 1993). Prescribed scenarios are used when it is not possible to record real human-human conversations due to some issues such as a privacy issue, or when there is no existing setting that matches a target dialog system. A scenario-based conversation is less natural since the goal of the conversation is not the participants own goal but is given in the scenario along with other constraints. The participants may not have real motivation to accomplish a task. Furthermore, the language used to describe the scenario can affect the user's choice of language which is known as a priming effect (Dybkjær et al., 1995). How well scenario-based conversations cover the domain also depended on a set of scenarios chosen. Nevertheless, scenario-based conversations still provide useful information about the characteristics of a task and the interaction between dialog participants. Yankelovich (1997) used pre-design studies to collect human-human conversations in four task-oriented domains. Conversations in one of the domains were recorded when the participants performed a task in a real situation while in other three domains prescribed scenarios were used. In his experiments, the analysis of the conversations collected from the pre-design studies, which took place prior to any system design, revealed users' perspectives of a task that might not be foreseen by dialog system developers and helped reduce major design problems. The main drawback of the method that elicits dialog system requirements from the analysis of human-human conversations is that the language used in those conversations is rich and unrestricted while the language that a dialog system can support is more limited.

To observe a conversation between a human and a dialog system before such a system is actually created, the simulation of the system is required. A *Wizard-of-Oz* (*WOZ*) method is commonly used to create simulated human-machine conversations. In this method, a human (a *wizard* or an experimenter) plays a role of a dialog system and responds to a user using a synthesized voice. The user is made to believe that he or she is interacting with a computer system. An example of the WOZ method can be found in the work by Bangalore, Fabbrizio et al. (2006). In this work, the data collected by the WOZ method was analyzed by a user experience engineer to determine dialog system specifications and its functionality. However, there are several concerns regarding a WOZ procedure (Churcher et al., 1997). For example, it is difficult for a human wizard to behave exactly like a dialog system, which has limited communication capability, and simulate speech recognition and understanding problems that may occur in a real system. The behavior of each wizard may also affect how a user interacts with the simulated system. To resolve this problem, a prototype system with limited functionality is used to collect real human-machine conversations. The system that has more completed functionality is then developed based on the analysis of this initial human-machine data.

The prototype of the CMU Communicator system was designed based on the analysis of both human-human data and simulated human-machined data (WOZ data) (Eskenazi et al., 1999). This prototype system was then used to collect real human-machine dialogs to further improve the automatic travel planning system.

As we can see from the discussion above, the analysis of in-domain conversations both human-human and human-machine conversations plays an important role in the requirement specification and design of a task-oriented dialog system. However, the analysis has been done manually which makes the process expensive, subjective and probably inconsistent (Bangalore, Fabbrizio et al., 2006).

1.4 Overview of the proposed solution

This thesis aims at exploring the feasibility of using a machine learning approach to infer the domain-specific information required to build a task-oriented dialog system from in-domain conversations. The machine learning approach could potentially alleviate the bottleneck that occurs in the conventional dialog system development process, where the domain-specific information is identified manually, and reduce human effort in developing a dialog system in a new task-oriented domain. In this section, I will first define the scope of this thesis work, and then describe the overview of the proposed approach.

Since the knowledge engineering process occurs before the first prototype system is created, a collection of recorded conversations between human participants that perform the same task as a target dialog system becomes a main resource. In many task-oriented domains, collections of human-human conversations already exist. One example is a collection of calls made to a help desk operator of the Pittsburgh Port Authority Transit system (Raux et al., 2003). Even if there is not one, in the case where we would like to replace one of the participants' roles with a dialog system, a corpus of human-human conversations can be collected quite easily. For instance, human-human conversations in a travel-planning domain can be collected by recording the conversations between travel agents and their clients (Eskenazi et al., 1999). In addition to a dialog corpus of a target domain, I am also interested in incorporating information from other knowledge sources that are already available to improve learning accuracy. One example of the existing resources is the WordNet lexical database.

There are some previous studies on the differences between human-human conversations and human-machine conversations (Dahlbäck et al., 1993; Hauptmann and Rudnicky, 1988; Jönsson and Dahlbäck, 1988). These studies show that the language a human uses to communicate with a computer is more constrained than the one he/she uses to communicate with another human participant. For example, the vocabulary size and the syntactic variations are smaller when compared to those of human-human conversations in the same domain. One explanation of this phenomenon is that humans adjust their language to accommodate the machine incomplete communication capability. However, the differences are in terms of the language used to communicate information. I assume that the structure of a task and domain keywords do not change with the communication ability of dialog participants, and that a corpus of human-human conversations is still a useful resource for acquiring the domain-specific information that will be used to create a dialog system.

The proposed machine learning approaches for inferring the domain-specific information from in-domain conversations are mainly unsupervised. When acquiring the domain-specific information for a new task-oriented domain, there is no annotated data available for training supervised learning algorithms as the target domain-specific information has not been specified and needs to be inferred from in-domain dialogs. For that reason, we have to rely on unsupervised learning algorithms. Acquiring the necessary domain knowledge from a set of human-human dialogs is considered a knowledge acquisition process and is carried out before a dialog system is created. This is contrasted with a dialog structure recognition process in which pre-specified dialog structure components are recognized as a dialog progresses. Although a supervised learning approach usually provides a more accurate result, it comes with the cost of manually labeled data. There are also some cases where an unsupervised learning approach performs better than a supervised one (Woszczyna and Waibel, 1994). In these cases, the unsupervised approach, which doesn’t get any influence from human annotation, better reflects the characteristics of the data. Furthermore, it is interesting to see how well an unsupervised learning approach can perform on the problem of domain knowledge acquisition and what would be its limitations.

In order to apply a machine learning approach to the problem of domain knowledge acquisition, two research problems have to be addressed: 1) specifying a target representation which captures the domain-specific information that a dialog system needs to have in order to support a task in that particular domain, and 2) developing a machine learning approach that infers the domain information captured by this representation from in-domain dialogs.

For the purpose of this dissertation research, a suitable domain knowledge representation should have all of the following properties:

- *Sufficiency:* capturing all domain-specific information required to build a dialog system in a new task-oriented domain
- *Generality:* being able to describe task-oriented dialogs in dissimilar domains and types
- *Learnability:* the representation can be identified by a machine learning algorithm from observable language behaviors in human-human conversations

In this thesis, I propose a *form-based dialog structure representation* as a target representation of the domain-specific information, which will be inferred from in-domain conversations, and claim that it has all of the required properties. The form-based dialog structure representation is a three-level structure of *task*, *sub-task* and *concept*. These components reflect the observable structure of a task-oriented conversation discussed in Section 1.2. Task and sub-task represent the decomposition of a complicated task while concepts are items of information (or domain keywords) that dialog participants have to communicate in order to achieve the conversation goal. A more formal definition of each component and examples of the component in various task-oriented domains are discussed in detail in Chapter 3**.** The proposed representation is based on the notion of *form*, a data representation used in the form-based dialog system architecture. The use of forms and a form-filling strategy in dialog systems was first introduced by Ferrieux and Sadek (1994) and has been adopted in many systems built on the form-based dialog system architecture. Forms are well understood by practitioners but in an informal way. Typically, a form corresponds to a database query form and contains items of information that are search criteria. This interpretation is specific to only information-accessing domains. In this thesis, a more generalized interpretation of the form representation is provided, so that it can be used to represent the structure of dialogs in other types of task-oriented domains as well.

In the following parts of this section, I will briefly discuss the properties of the form-based dialog structure representation and the approaches that are used to verify these properties. Since the form-based dialog structure representation is based on the data representation used in the form-based dialog system architecture, it captures all of the domain information required to build a form-based dialog system. Thus, the form-based representation is sufficient for representing task-oriented dialogs, at least in information-accessing domains, as demonstrated by the success of the systems that were implemented based on the form-based architecture. To verify its sufficiency for other types of task-oriented domains and its generality, six dissimilar task-oriented domains are analyzed.

Those six domains are air travel planning (information-accessing task), bus schedule inquiry (information-accessing task), map reading (problem-solving task), UAV flight simulation (command-and-control task), meeting and tutoring. These domains are chosen to cover dissimilar types of task-oriented domains. The choices of domains are also subjected to the availability of human-human data. *Dialog coverage*, which measures the percentage of dialog content that can be accounted for by the proposed dialog structure, is also used to verify the sufficiency of the form-based dialog structure representation.

In terms of learnability, since the components of the form-based dialog structure representation can be observed directly from a dialog as they reflect the observable structure of a task-oriented conversation discussed in Section 1.2, the form-based representation should be learnable through an unsupervised learning algorithm. The accuracy of the domain information obtained by the proposed machine learning algorithms is used to verify the learnability of the form-based representation. The proposed learning algorithms are described below. An additional evaluation, a human annotation experiment, is carried out to verify that the proposed form-based dialog structure representation can be understood and applied reliably by annotators other than the coding scheme developer. This evaluation also verifies the learnability of the form-based representation in terms of human learnability. High annotation scheme reliability suggests that the annotation scheme is concrete and unambiguous which also imply learnability. Annotation scheme reliability is assessed along two aspects, *reproducibility*, which measures the level of agreement among novice coders and, and *accuracy*, which measures the correctness of a novice coder's annotation when compared to an expert's annotation.

The form-based dialog structure representation also has another desirable property, a direct mapping between dialog structure components and dialog system components. By using an existing dialog system framework to describe the structure of a task-oriented conversation, the connections between the components of the form-based dialog structure representation and the components of the system that employs the representation become straightforward. This direction is opposite to many other research works that implement a dialog system from an existing dialog structure theory.

To make a dialog structure acquisition problem tractable, I divide the problem into two sub-problems: concept identification and form identification (a form is associated with a sub-task in the form-based dialog structure representation). Each sub-problem is handled separately. However, it should be kept in mind that these individual components are parts of the same dialog structure; therefore, information about one component may be useful for inferring another component. After each component can be acquired with acceptable accuracy, interaction between components should also be considered in the learning process. The accuracy of the acquired domain information is evaluated by comparing the learned dialog structure components to the reference components identified by a domain expert.

To identify a set of domain concepts, a learning algorithm has to identify instances of concepts and group the ones that belong to the same concept type together. Since a list of concept types in a given domain is not pre-specified but will be explored from data, the concept identification problem is different from a classification problem, for example, named entity extraction. In the classification problem, a word or a group of words is classified as one of the predefined roles such as person and organization. Two concept clustering approaches are investigated in this thesis, statistical clustering and knowledge-based clustering. By assuming that words that have similar meaning tend to occur in similar context, a statistical clustering approach groups concept words together based on their distributional similarity. Two statistical clustering algorithms, mutual information-based clustering and Kullback-Liebler-based clustering, are compared. Both algorithms are an agglomerative hierarchical clustering approach (or a bottom-up approach); however, they use different heuristics to determine the similarity between words or groups of words. Automatic stopping criteria based on the measures available during the clustering process are also proposed for both approaches. For knowledge-based clustering, the proposed approach utilizes semantic information stored in the WordNet lexical database and groups concept words together based on their hypernyms.

The goal of the second learning problem, form identification, is to determine different types of forms and their associated slots that a dialog system needs to know in order to perform a task. Since a form represents a segment of a dialog that corresponds to a sub-task, identifying the sub-tasks in a set of in-domain dialogs can help determine a set of forms. The proposed solution is as follows: first segment a dialog into a sequence of sub-tasks, then group the sub-tasks that are associated with the same form type into a cluster. By analyzing the concepts contained in each cluster, a set of slots that is associated with each form can be obtained. Two unsupervised discourse segmentation approaches are investigated: a TextTiling algorithm and a Hidden Markov Model. Both approaches, while performing well with expository text, require some modifications when they are applied to a fine-grained segmentation problem of spoken dialogs. The proposed modifications include: a data-driven stop word list, a distance weight and an appropriate context size. After segmenting all dialogs into sequences of sub-tasks, the bisecting *K*-

means clustering algorithm is used to group the dialog segments that belong to the same type of sub-task together as they represent the same form type. The bisecting *K*-means algorithm is an unsupervised clustering algorithm that utilizes cosine similarity between dialog segments in order to assign the segments into clusters. Information from concept annotation is included as an optional feature for both dialog segmentation and sub-task clustering algorithms to see how information from another dialog structure component affects the learning accuracy.

1.5 Thesis statement

This thesis investigates how to infer, from a corpus of task-oriented human-human conversations, the domain-specific information that a dialog system needs to have in order to support a task. The required domain information is clearly reflected in the structure of a dialog as dialog participants exchange pieces of information to perform a task, and for a complex task, decompose it into a set of sub-tasks which they can pursue one at a time. For the domains that exhibit this information-exchanging characteristic, when the domain-specific information is clearly expressed in a dialog, it can be automatically acquired from the associated language behaviors through unsupervised machine learning approaches.

1.6 Thesis organization

The following is an outline of the remainder of this thesis.

- Chapter 2**:** *Literature Review*. This chapter summarizes several well-known discourse structure representations and prior research works that attempted to identify those structures from data. The chapter also discusses the limitations of the existing discourse structure representations if they were to be used in the context of this dissertation.
- Chapter 3**:** *Form-based Dialog Structure Representation.* This chapter describes the proposed form-based dialog structure representation and argues that it has all of the required properties. Examples on how to model the structure of a dialog with the proposed representation in six disparate types of task-oriented domains are given. The chapter also discusses the limitations of the proposed structure and compares it to existing dialog structures.
- Chapter 4**:** *Form-based Dialog Structure Representation Evaluation.* This chapter describes two evaluation procedures: dialog coverage, which measures the percentage of dialog content that can be accounted for by the

proposed structure, and annotation scheme reliability, which assesses annotation agreement among novice coders to verify that the proposed form-based dialog structure can be understood and applied by annotators other than the coding scheme developer.

- Chapter 5**:** *Concept Identification and Clustering*. In this chapter, approaches for identifying a set of domain concepts are described along with their evaluations.
- Chapter 6**:** *Form Identification*. This chapter consists of two parts which correspond to the two major steps used to identify different form types and their associated slots. The first part describes dialog segmentation algorithms. The second part describes sub-task clustering algorithms along with the analysis of the potential slots for each type of form. A series of experiments used to evaluate these learning algorithms are reported.
- Chapter 7**:** *Conclusion*. This chapter summarizes all the findings and discuses directions for future research

Chapter 2

Literature Review

This chapter reviews existing research works in two areas: discourse structure representation, and data-driven approach to dialog structure modeling. These two areas are related to the two research problems that have to be solved in order to achieve the goal of this dissertation which is to infer the domain-specific information required to build a task-oriented dialog system from in-domain conversations through a machine learning approach. The two research problems that have to be solved are 1) specifying a dialog structure representation that is suitable for representing the required domain-specific information, and 2) developing a machine learning approach that infers the domain information captured by this representation from in-domain dialogs. Throughout this thesis, the terms *discourse* and *dialog* can be used interchangeably. Nevertheless, in some specific discussions, the term discourse may have a boarder interpretation that includes both monologs and dialogs.

This chapter is organized as follows: Section 2.1 summarizes well-known discourse structure theories and models. Section 2.2 discusses the learning approaches that have been used to identify a structure of a dialog from data.

2.1 Discourse structure representations

Discourse structure modeling has been a topic of interest for several decades. Many theories and models have been proposed to explain a structure of a human-human conversation. These dialog structure models (or theories) focus on different aspects of dialogs depended on the purpose of the models and the assumptions they made about human-human conversations. In addition to linguistics, the ideas behind discourse structure theories and models were also influenced by many other fields of study including psychology, sociology, philosophy, and computer science.

In the early days, the research in the area of discourse structure modeling focused mainly on developing a theory that facilitates the interpretation of discourse meaning that goes beyond the level of an individual utterance. These discourse structure theories were derived mainly from linguistic and psychological point of views, and were aimed preliminary at describing discourse phenomena in both monologs and dialogs with the proposed structure. Recent works on discourse structure modeling are more engineering-oriented. Practical issues such as predictability of each structure component and a computational model that can represent the proposed dialog structure in an automate system have been addressed when developing a discourse structure model. Many of these representations are derived from the analysis of recorded human-human conversations, and thus receive more influence from real data than linguistic and psychological theories.

In this section, I will review some of the works that are well known in the field or have been applied in dialog system implementation. Additional reviews on discourse structure representation can be found in many survey articles including the chapter 6 of Cole et al.'s (1997) article, Grosz et al.'s (1989) article which focuses on discourse structure for natural language understanding, and Moore and Wiemer-Hastings' (2003) article which focuses on discourse structure for natural language generation.

Most discourse structure models agree that a discourse has a compositional structure (Grosz et al., 1989; Moore and Wiemer-Hastings, 2003). That is, a discourse can be divided into coherent segments. Moreover, these segments also possess some relations among one another. Discourse segments and their relations constitute the structure of the discourse. However, what discourse segments and their relations represent can be different depended on the aspect of a discourse that each model emphasizes. A discourse can be viewed from two different perspectives: an informational perspective and an intentional perspective. This categorization is similar to the informational-intentional distinction discussed in the chapter 6 of Cole et al.'s (1997) article and also in (Hobbs, 1996; Moore and Pollack, 1992)

1. *Informational perspective* (or *content-based perspective*) captures the actual content being conveyed in a discourse. The content of the discourse can be modeled by its surface representation, such as the actual entity that was mentioned, or by its semantic representation.

2. *Intentional perspective* captures a speaker's intention behind each utterance (i.e. why it was said) and the overall goal of a discourse.

Some discourse structure representations may capture both the informational and the intentional aspects of a discourse, but only one aspect is emphasized. Based on the emphasized aspect, discourse structure representations can be broadly categorized into two groups: *informational-oriented* discourse structures and *intentional-oriented* discourse structures. After reviewing each group of discourse structure representations in Section 2.1.1 and Section 2.1.2 respectively, all discourse structure representations are compared in Section 2.1.3. In Section 2.1.3, I will also discuss the appropriateness of these representations if they were to be used in the context of this dissertation.

2.1.1 Informational-oriented discourse structures

2.1.1.1 Discourse Representation Theory (DRT)

Discourse Representation Theory (*DRT*) is a formal semantic model which focuses on the semantic truth conditions of a discourse and aims primarily for discourse understanding. The theory was first introduced by Kamp (1981) and further developed by Kamp and Reyle (1993). DRT uses a logical language to represent the meaning of text similar to a first-order predicate logic; however, the logical representation in DRT is extended from the level of a sentence to the level of a discourse by including the representation of the context. The semantic representation of a discourse in DRT is called a *Discourse Representation Structure* (DRS). The meaning of a given text is derived on a sentence-by-sentence basis. Semantic interpretation rules are used to transform the syntactic structure of each sentence to the semantic one. The interpretation of each sentence is made in the context of preceding sentences which is represented by the current DRS. The result of this interpretation is then used to update the DRS. The advantage of this approach is that the semantic representation of a discourse is built up from the contents of the discourse alone without bringing in external information. DRT provides a computation framework to resolve some linguistic issues, such as anaphora resolution and quantifier scoping, through predicate calculus which can be implement using LISP or PROLOG. Although, DRT provides a representation for a discourse, it focuses more on describing the truth conditions of the discourse rather than its compositional structure. Most of DRT mechanisms also focus on sentence level processing without taking into account the relationship between sentences.

2.1.1.2 The Linguistic Discourse Model (LDM)

The Linguistic Discourse Model (*LDM*) (Polanyi, 1996) provides a framework for discourse interpretation based on the linguistic theory of discourse structure that has been developed by Polanyi and her colleagues since 1984. In the LDM framework, both structural relations and semantic accessibility relations (relations among contextual categories) play important roles in discourse interpretation. A structural description of a discourse is represented by a *Discourse Parse Tree* (DPT). Each leaf node in the DPT is a *Discourse Constituent Unit* (DCU), a semantically motivated discourse unit that expresses a single event or state of affairs in a discourse world. A DCU is equivalent to a single clause or a single phonological phrase and is often marked by *discourse operators* (or discourse markers). These discourse operators also provide information about the relations among the DCUs.

A DPT can be constructed from DCUs using a discourse grammar. Polanyi argued that for most of the cases, simple context-free rewrite rules are sufficient to describe the structure of a discourse. The discourse is processed on a DCU-by-DCU basis. The relation between a new DCU and its immediately preceding DCU determines how the new DCU will be attached to the DPT. This relation also determines how the semantic interpretation of the new DCU will be combined with the current semantic representation of the discourse. A semantic representation in the LDM captures both propositional content and discourse contexts, which are modeled by a hierarchy of contextual categories (*interaction*, *speech event*, *genre unit*, *modality*, *polarity*, and *point of view*). The semantic representation in the LDM is an extension of the discourse representation structure (DRS) used in Discourse Representation Theory (DRT) discussed in Section 2.1.1.1. However, while DRT emphasizes discourse referents, the LDM emphasizes the setting and resetting of discourse contexts.

Similar to DRT, the LDM focuses on a semantic representation of a discourse which is described in terms of the truth conditions. The LDM also describes a structural representation of a discourse; however, a discourse parse tree is influenced by sentential syntax rather than a task structure or discourse goals.

2.1.1.3 Segmented Discourse Representation Theory (SDRT)

Segmented Discourse Representation Theory (*SDRT*) (Asher, 1993) is the extension of Discourse Representation Theory (DRT), discussed in Section 2.1.1.1, that takes into account the structure of a discourse when combining the semantic representation of a new sentence into the overall semantic representation of the discourse. Instead of simply merging the sentence-level representation with the current Discourse Representation Structure (DRS) and creating a larger DRS as in DRT, SDRT uses a discourse relation between the new sentence and its previous sentence to determine how the semantic representation of the new sentence should be combined with the analysis of previous sentences in the overall structural semantic representation.

From the influence of DRT and the analysis of anaphora, SDRT views the structure of a discourse from a semantic perspective and can be considered as a semantic theory of discourse structure. A unit of a discourse (or a discourse segment) is defined at the level of proposition and is equivalent to a simplest form of a DRS in DRT. A set of discourse segments, namely a set of DRSs, and a set of discourse relations between these DRSs determine the structure of a discourse. This discourse structure, called a *Segmented Discourse Representation Structure* (*Segmented DRS* or, *SDRS*), is imposed on the top of the semantic representation in DRS. Since SDRS is defined recursively, it many contain another SDRS, and thus constitutes a hierarchical discourse structure.

There are two types of relations: *structural relation* and *non-structural relation* (or *semantic relation*). Structural relations specify how a discourse is segmented and how the segments are organized. For examples, Continuation indicates that a new sentence and its previous sentence are in the same discourse segment while Elaboration indicates that a new sentence is in the discourse segment that is dominated by the discourse segment of its previous sentence. Nevertheless, structural relations do not affect the semantic truth conditions of discourse content. Non-structural relations such as CAUSE, on the other hand, have some implications on the semantic truth conditions of the discourse content. Satisfaction conditions of these relations impose constraints on the locations in the current SDRS that a new sentence can be attached to. A list of discourse relations are open; but for one specific set of data, the number of relations should be small.

The overall structure of a discourse is created on a sentence-by-sentence basis similar to DRT. In order to attach the DRS of a new sentence to the current SDRS, both an attachment point and the discourse relation between the new DRS and its preceding sentence have to be identified. The discourse relation can be inferred from various sources of information including cue words, available attachment points, the content of the current SDRS and the new DRS, and satisfaction conditions of possible relations.

SDRS is quite similar to the Linguistic Discourse Model (LDM) discussed in Section 2.1.1.2. However, SDRS puts more emphasis on the semantic aspect of a discourse structure while the LDM puts more emphasis on the syntactic and structural aspects of the structure. This difference is reflected in the choices of discourse relations and their effects on discourse processing. The relations in LDM mostly affect the structure of a discourse while some relations in SDRS also affect the semantic representation of the discourse.

2.1.1.4 Rhetorical Structure Theory (RST)

Rhetorical Structure Theory (*RST*) (Mann and Thompson, 1988) was originally developed from the analysis of carefully prepared text from various sources. The theory explains a structure of a discourse in terms of relations between its parts. The assumption behind RST is that every part of coherent text has a reason for its presence. RST provides a rich set of coherence relations, yet principally open, that describes a role that one text span (a *satellite*) has with respect to another text span (a *nucleus*). These coherence relations were defined based on functional and semantic criteria rather than syntactic ones. There are two types of relations: a *subject matter* relation and a *presentational* relation. A subject matter relation (e.g. elaboration and cause) is intended for a reader to recognize the relation while a presentational relation (e.g. Motivation and Justify) is intended to increase some inclination in the reader. From the two aspects of a discourse discussed earlier, a subject matter relation is considered informational as the relation itself has to be recognized in order to understand the discourse while a presentational relation is considered intentional as it affects the reader's belief not the meaning of the discourse.

Since a text span is roughly defined as any uninterrupted linear interval of text, one can create a hierarchical structure of text by identifying coherence relations between all of the compositions of a given discourse (e.g. between sentences, groups of sentences and paragraphs). Such a structure is called a *rhetorical structure tree* or a *discourse tree*. Recently, Taboada and Mann (2006) reviewed and responded to the issues that have been addressed on theoretical aspects of RST. In terms of learning, Marcu (1999) proposed a rhetorical parsing algorithm that learns to construct a rhetorical structure of text from annotated data using a decision tree.

RST has been used mainly for text generation such as automatic summarization. To apply RST to dialogs several modifications are required in order to handle dialog-specific behaviors. Stent (2000) captured the collaboration between participants in task-oriented dialogs by introducing a new set of relations that describe adjacent pairs such as question-response and proposal-accept. However, like other relations, adjacency pairs emphasize speakers' rhetorical goals rather than task goals. To capture task-specific structural patterns, the notion of *schema* (e.g. make-plan and describe-situation) was added to the annotation scheme.

2.1.2 Intentional-oriented discourse structures

2.1.2.1 Speech act theory

Speech act theory has its root in the field of philosophy of language from the work of a philosopher J. A. Austin and his follower J. R. Searle. This theory focuses on the function of language that goes beyond the level of semantics (i.e. the truth value of a proposition). Speech act theory analyzes the role of an utterance with respect to the intention of a speaker (the illocutionary force) and the effect on a listener (the perlocutionary effect), and thus introduces pragmatics to the field of discourse structure modeling. Several categories of speech acts have been proposed; however, the one that has the strongest influence on the set of *dialog acts*[1] used in many dialog systems is Searle's taxonomy of illocutionary acts (Searle, 1975). Searle argued that a number of basic categories of intentions behind the use of language is definite and proposed five categories of illocutionary acts which are *assertives* (a speaker conveys the belief that something is being the case), *directives* (a speaker attempts to get a hearer to do something), *commissives* (a speaker commits to do something in the future), *expressives* (a speaker expresses his/her feelings), and *declarations* (a speaker changes the state of the world by saying the utterance). Each utterance may contain more than one illocutionary act. Many researchers have modified the speech act taxonomy to better suite their tasks by adding more domain-specific acts. Alexandersson et al. (1995) extended Searle's speech act taxonomy to a set of 17 *dialog acts* that describes appointment scheduling conversations in the VERBMOBIL speech-to-speech translation system.

Speech act theory does not describe the overall structure of a discourse but focuses only on the level of an utterance. Nevertheless, a speaker's intention captured by a speech act is also a key component in many other theories that use an utterance as a discourse structure unit including a dialog grammar, a plan-based model and the theory of conversation acts. The details of these theories are discussed below.

2.1.2.2 Dialog Act Markup in Several Layers (DAMSL)

The *DAMSL* annotation scheme (Core and Allen, 1997) was developed from speech act theory (Searle, 1975) discussed in the previous section. However, instead of using a single label to capture an utterance's purpose as in speech act theory, DAMSL uses multiple labels in multiple layers to describe the utterance's function in various aspects. The DAMSL annotation scheme consists of three orthogonal layers: *Forward Communicative Functions*, *Backward Communicative Functions*, and *Utterance Features*. The Forward Communicative Functions contain a taxonomy similar to the actions in speech act theory. The Backward Communicative Functions contain a set of labels that indicates the relation between the current utterance and the previous ones such as agreement and answer. The Utterance Features describe the content and form of an utterance.

DAMSL was developed by Multiparty Discourse Group as an annotation guideline for task-oriented conversations in general. The communicative acts defined in DAMSL are primitive communicative actions that are common in various task-oriented domains.

[1] The term *dialog act* may also be used interchangeably for the term *speech act*.

These communicative acts can be extended to include domain-specific acts as shown in Meeting Recorder Dialog Act (MRDA) (Dhillon et al., 2004) and the dialog move taxonomy for tutorial dialogs (Tsovaltzi and Karagjosová, 2004). The DAMSL annotation scheme can also be augmented with additional layers that describe an utterance's functions in other aspects. To better describe reasoning and problem-solving processes in problem-solving conversations, two additional layers were introduced in the COCONUT project (Eugenio et al., 1998). The *Topic* layer describes the content of an utterance with domain-specific tags while the *SurfaceFeatures* capture syntactic features of the utterance such as tense and subject. Hardy et al. (2003) added the *semantic* layer to capture domain-related information enclosed in each utterance.

Since DAMSL was developed from speech act theory, it too does not describe an overall structure of a discourse but focuses only on the level of an utterance. Each utterance is described in isolation. A relation between utterances is not captured except for the link to the antecedent, the previous utterance being responded to by the current utterance, provided by Backward Communicative Functions. A structural relation between groups of utterances is not specified. Nevertheless, DAMSL is widely used in many dialog systems to aid the interpretation of user utterances similar to other extensions of speech act theory. Furthermore, it has been shown that the DAMSL tagset and its extension can be automatically recognized with acceptable accuracy (Jurafsky et al., 1997; Stolcke et al., 2000).

2.1.2.3 Dialog grammars

The idea of a *dialog grammar* is based on the observation that a conversation contains regular patterns. The most prominent pattern is known as an *adjacency pair* such as a question/answer pair. In a collaborative conversation, we can assume that the succeeding utterance will follow the initiative set by the preceding utterance. For example, we can expect that a question will be followed by an answer. Generally, each pattern is a sequence of utterances in which the first utterance of the sequence (or an *initiation*) creates a discourse expectation that will be fulfilled by subsequent utterances (or *responses*). These sequences are then built up into larger patterns in a dialog. Regular patterns in a dialog are hierarchical and can be expressed by a grammar. Each grammar rule specifies how a dialog or a dialog segment is decomposed into smaller units. Usually, the smallest unit of a dialog, or a terminal node in a dialog grammar, is represented by a dialog act. The next level non-terminal node is an adjacency pair which corresponds to a certain sequence of dialog acts. Non-terminal nodes higher up in the hierarchy could be motivated by the characteristics of a conversation. For example, in a task-oriented conversation, some non-terminal may correspond to the sub-goals. By describing a dialog using a grammar, the structure of a dialog can be obtained by parsing the dialog similar to sentence structure parsing. Unlike some other theories that can be applied to any type of discourse, this theory is specific to dialogs.

One well-known dialog grammar model is a five-level structure proposed by Sinclair and Coulthard (1975) from the analysis of the language used by teachers and students. The five levels are *lesson*, *transaction*, *exchange*, *move*, and *act*. The largest unit of a classroom discourse is a lesson. A lesson is a collection of transactions; each of them corresponds to one segment of a dialog that has a specific purpose and contributes toward the goal of the conversation. A transaction is equivalent to a *discourse segment* in Grosz and Sidner's Theory of discourse structure (Grosz and Sidner, 1986) discussed in Section 2.1.2.6. A transaction in turn consists of a set of related exchanges, a set of initiation-response sub-dialogs. An exchange is the most apparent pattern in a dialog and can be considered as a more general case of a question-answer pair. It consists of an initiation, a response, and a feedback, which are the categories of moves. A move is the smallest free unit that composes of the smallest dialog units called acts. These acts are also known as *dialog acts* and usually are an extended set of the original speech acts (Searle, 1975).

The HCRC dialog structure (Carletta et al., 1996) adopted the structure proposed by Sinclair and Coulthard (1975), but used only the three middle levels, to describe the phenomena in problem-solving conversations in a map reading domain. At the highest level of the HCRC hierarchical structure, a dialog is divided into *transactions*. Each transaction is a sub-dialog that corresponds to a major step of a task. A transaction is made of a sequence of *conversational games* or initiation-response exchanges. Each exchange consists of an initiation and a sequence of responses that fulfills a discourse expectation set by an initiation. Each initiation or response is called a *move* and corresponds to an utterance or a part of an utterance. Lewin et al. (1993) incorporated dialog move recognition in the dialog manager of the automatic route-planning system to predict the move of another conversation participant. In addition, the corpora annotated with the HCRC dialog structure were used to study various language phenomena such as intonation and effects of communication conditions. Please refer to the references in the conclusion of Carletta et al.'s (1997) article for more detail.

In summary, a dialog grammar describes a tree structure of a dialog that has dialog acts as terminal nodes and larger dialog segments such as initiation-response exchanges as non-terminal nodes. The dialog grammar can be used to prescribe acceptable dialogs in a given domain. The grammar rules can also be used to predict the next element in a conversation. The dialog grammar is employed in some dialog systems to predict subsequent user utterances and guide a conversation. One example is the rules of conversations in SUNDIAL ESPRIT project (Bilange, 1991). The limitation of the dialog grammar is that it might be too restrict to describe complicated dialogs. Since the dialog grammar uses grammar rules to specify acceptable dialogs, it is quite difficult to generate a set of rules that covers all possible variations of conversations in a complex domain.

2.1.2.4 Plan-based models

In a *plan-based model*, a conversation is perceived as a plan that dialog participants execute in order to achieve some goals. A plan consists of a sequence of operators that transforms an initial world state to a goal state. In a conversation, a speech act is an operator that produces an utterance and causes some effect on a hearer and the state of the hearer's world, such as modifying the hearer's belief. A plan-based model also describes how speaker intentions captured by speech acts fit together in the conversation and how they relate to the conversation goal. Examples of plan-based models can be found in the works of Carberry (1990), Cohen and Perrault (1979), and Pollack (1992).

During a conversation, a participant attempts to recognize the plan of another participant in order to make an appropriate response. Allen and Perrault (1980) described a computational model that infers another participant's plan from observed actions. An *action* (or an operator) is defined in terms of *preconditions*, criteria that have to be achieved before executing the action, and *effects*, how the hearer's world model changes after executing the action. The plan inference process can be done by: 1) given expected goals, searching for a plan that includes observed actions, or 2) using inference rules to infer a goal from observed actions. Partial plans are rated by how well-form the plans are in the given context and how well they conform to the expectation. Since a plan-based model describes the relations between utterances and a conversation goal, its principle can be applied to dialog control or dialog management. The plan-based approach to dialog has been implemented in many complex dialog systems such as the VERBMOBIL speech-to-speech translation system (Alexandersson, 1995).

Traditional plan-based models are quite rigid as they make rather strong assumptions about the nature of a plan, its elements and the environment in which the plan will be executed. For example, they assume that there is no change in the world between the time of planning and the time of execution, and that dialog participants' beliefs persist. However, those assumptions are not practical in real situations. Several augmented plan-based models have been proposed to address these issues.

The traditional models work well for a dialog that follows the task structure closely, but have a problem accounting for some types of sub-dialogs (e.g. clarification, correction, and topic change) since the models only allow an utterance to describe a step in a plan. Litman and Allen (1987) introduced *discourse plans* to describe various ways an utterance can relate to a discourse topic and distinguished them from *domain plans* that are actually used to model the topics (plans in a traditional plan-based model). Discourse plans explicitly represent discourse intentions and incorporate the knowledge about a discourse into a plan-based model in addition to the knowledge a domain captured by traditional domain plans.

A tripartite model (Lambert and Carberry, 1991) further differentiates discourse plans into *problem-solving plans* and *communicative plans*. The relationships among the three types of plans are organized into a hierarchical dialog model with discourse plans (communicative plans) at the lowest level, problem-solving plans at the middle level, and domain plans at the highest level. The actions in the lower level plans contribute toward the actions in the higher level plans. Nevertheless, actions in all levels can be recognized incrementally as the tree structure is allowed to grow from both the root and the leaves. A tripartite model provides a finer-grained differentiation among different types of user intentions and allows a different processing to be applied to each type of plan.

In a negotiation sub-dialog, dialog participants may change their beliefs as the dialog progresses which conflicts with a persistent belief assumption made by the traditional plan-based models. To handle the changes in beliefs, a multi-strength belief model and *acceptance actions*, were added to the tripartite plan-based model (Lambert and Carberry, 1992). An acceptance action, which is included in a discourse plan, addresses the understandability, believability, or relevance of a particular proposition communicated by the participants. The plan-based model combines multiple knowledge sources including linguistic, contextual, and world knowledge to recognize the changes in beliefs. Rosé (1995) extended the tripartite model further in order to capture multi-threads of negotiations in the scheduling domain. The changes in beliefs occur in negotiation sub-dialogs suggest that dialog participants need to have shared beliefs in order to collaborate on a task. The notions of *mutual belief* and *shared plan* are discussed in a collaborative planning model in Section 2.1.2.6.

Rather than assume that an environment is static as in the traditional plan-based models, the BDI (Belief, Desire, and Intention) architecture (Bratman et al., 1988; Pollack, 1992) allows a plan to be executed in a dynamic environment where it may change in the way that makes the plan invalid. The BDI model, also known as IRMA (the

Intelligent, Resource-Bounded Machine Architecture), is based on the idea of practical reasoning developed by Bratman (1987). Beliefs are uninstantiated plans while desires are a participant's goals. Intentions are steps in the plan that the participant has committed but not yet acted on. To handle the change in the environment, the participant is allowed to make changes to the plan that he/she has already committed. At each step in the plan, the participant can choose to continue with the current intention or adopt one of the new options arise from the change in the environment. The new intention may better suite the new beliefs and goals that are the results of the change. However, as practical reasoning poses the constraint on the amount of resources available for planning, the decision at each step can be sub-optimal. The BDI model was adopted in the TRAINS system (Ferguson et al., 1996), a dialog system that helps a manager solve a routing problem in a transportation domain. A reactive planner, such as the one developed by Georgeff and Ingrand (1989), is also implemented on the basis of practical reasoning and the BDI model. Even though reactive planning was originally developed for real-time control systems, it can also be used in a variety of other domains such as tutoring (Freedman, 2000).

The ability to represent complicated conversations of a plan-based model comes with the cost of a complex dialog structure. In order to apply the plan-based model to a particular task-oriented domain, the following components have to be specified: beliefs and goals of the participants, a plan library, and plan elements (e.g. actions and their preconditions and effects). The complex structure also leads to a complicated plan recognition process. Other drawbacks of the plan-based model are mentioned in the chapter 6 of Cole et al.'s (1997) article.

2.1.2.5 The theory of conversation acts

The theory of conversation acts (Traum and Hinkelman, 1992) views a dialog as composed of fine-grained actions similar to speech act theory (Searle, 1975) discussed in Section 2.1.2.1. However, several extensions were introduced in the theory of conversation acts to make it better accounts for the structure of a spoken discourse. The theory of conversation acts models the conversation as a collection of joint speaker-hearer actions instead of single agent actions. This eliminates a mutual understanding assumption among conversation participants and makes grounding actions more explicit. The extensions include three levels of actions in additional to the core speech act level. The theory of conversation acts describes four levels of actions necessary for maintaining the coherence and content of the conversation. These four levels are *turn-taking acts*, *grounding acts*, *core speech acts,* and *argumentation acts*. These levels are typically realized by larger segments respectively in a dialog; however, the four-level representation of conversation acts is not hierarchical as each dialog act level is independent from each other and concerns a distinct aspect of the dialog. Turn-taking acts model the participants' control over a speaking channel while and grounding acts capture mutual understanding among the participants. Core speech acts are similar to the traditional speech acts and operate at the level of an utterance. The argumentation acts level accounts for the structure of the dialog above the level of an utterance. Argumentation acts capture the purposes of discourse segments and can be built up into a hierarchy of argumentation acts. At the top levels of the hierarchy, argumentation acts resemble tasks and sub-tasks in a task structure while, at the lower levels, they are similar to rhetorical relations (Mann and Thompson, 1988) and adjacency pairs.

Since each level of conversation acts captures distinct dialog information, they are employed independently from each other in a dialog system. Turn-taking acts may not be necessary in two-party conversations, but are more crucial in multi-party conversations while argumentation acts are important in a dialog system that involves complex planning. The theory of conversation acts emphasizes more on coordinated activities in the conversation, such as turn-taking and grounding, rather than the domain information communicated. Among the four levels of conversation acts, the argumentation act level is the one that captures the overall structure of a conversation similar to the structure of a task. Nevertheless, how to recognize argumentation acts and use them in a dialog system was only briefly discussed in the theory where the authors suggested the use of cue words together with the knowledge about discourse, language, and the domain for argumentation act recognition.

2.1.2.6 Grosz and Sidner's Theory of discourse structure

Grosz and Sidner's Theory of discourse structure (GST) (Grosz and Sidner, 1986) provides a framework for interpreting the meaning of an utterance in discourse context and for understanding discourse phenomena such as interruption based on the idea that a proper account of a discourse structure provides the basis for the interpretation of discourse meaning. GST models a structure of a discourse based on the concepts of discourse unit and discourse coherence. The proposed structure is composed of three components: *linguistic structure* (the structure of utterances in a discourse), *intentional structure* (the structure of purposes), and *attentional state* (the state of focus of attention).

The linguistic structure captures how utterances in a discourse are aggregated into discourse units. A discourse unit or a *discourse segment* is defined as a sequence of utterances which fulfills a certain function with respect to the overall goal of the discourse. The intention underlies each discourse segment is called the *discourse segment purpose*. We could also say that a discourse segment is defined based on dialog participants' intention. The second component, the intentional structure, models relationships between discourse segment purposes, and thus captures discourse coherence. These relationships are structural relations between intentions rather than relations between discourse segments as in Rhetorical Structure Theory (Mann and Thompson, 1988) discussed in Section 2.1.1.4. Therefore, the number of relations is smaller and the relations are also simpler. The last component, the attentional state, contains objects, properties, relations, and the purpose of the discourse segment that receives the focus of attention from discourse participants at any given point of the discourse. The attentional state models the participants' focus of attention during a conversation via *focusing structure* which uses the information from the intentional structure to determine a discourse segment that receives the focus of attention as the conversation progresses. The three components together supply contextual information necessary for the interpretation of utterances in discourse context.

GST describes an abstract model of discourse structures. To construct a computational model based on this theory the following problems need to be solved: discourse segmentation, and the recognition of discourse segment purposes and the relationships between them. Grosz and Sidner discussed some of these processing issues and suggested the use of cue phrases, utterance-level intention, and the knowledge about domain actions and objects to resolve the problems; however, no concrete implementation of the structure was proposed. Grosz and Sidner (1990) argued that a computational theory for recognizing discourse segment purposes and the intentional structure depends on the underlying theory of intention, action and plan and proposed *SharedPlans*, a model of collaborative planning that takes into account mutual beliefs and multi-agent actions in addition to a mental state model of a single-agent plan. This model also provides a framework for modeling the intentional structure. Lochbaum (1998) implemented a computational model that can recognize the intentional structure, and used it in discourse processing. The extension of SharedPlans that can handle more complicated situations in collaborative planning was proposed by Grosz and Kraus (1996). The model of SharedPlans emphasizes discourse-level intentions while a single-agent plan (Cohen and Perrault, 1979) only concerns with utterance-level intentions. Plan-based models are discussed in more detail in Section 2.1.2.4. The SharedPlan formalism was adopted in the COLLAGEN framework (Rich et al., 2001) which provides an intelligent user interface in various application domains.

2.1.3 A comparison of existing discourse structure representations

In this section, I will first compare all the discourse structure representations reviewed in Section 2.1.1 and Section 2.1.2. Then, I will discuss the appropriateness of these representations if they were to be used in the context of this dissertation. The following table summarizes the characteristics of each discourse structure model along the two aspects, informational and intentional, discussed earlier. The forth column of the table describes how each discourse structure model describes a compositional structure of a discourse. The reference given in the first column refers to the primary work of each model.

The first group of discourse structures in Table 2.1, informational-oriented discourse structures, includes Discourse Representation Theory (DRT), the Linguistic Discourse Model (LDM), Segmented Discourse Representation Theory (SDRT) and Rhetorical Structure Theory (RST). For the informational perspective, DRT, LDM and SDRT capture the content of a discourse with a semantic representation (a first-order predicate logic) while RST focuses more on the relations between discourse segments rather than their content. In terms of the informational perspective, none of DRT, LDM and SDRT explicitly models the participant’s intentions. For RST, even though it captures both informational perspective and intentional perspective of a discourse through subject matter relations and presentational relations respectively, the theory focuses more on the informational perspective (Moore and Pollack, 1992). DRT, LDM, and SDRT differ from each other mainly in their compositional structures, i.e. how the structure of a discourse affects its semantic representation. DRT does not consider discourse relations among sentences when creating a discourse-level semantic representation from sentence-level semantic representations. In LDM, the discourse relations only affect the compositional structure of the discourse-level semantic representation while, in SDRS, the discourse relations also affect the content of the semantic representation.

Discourse structure model	Informational perspective	Intentional perspective	Compositional structure	Remarks
Informational-oriented				
Discourse Representation Theory (DRT) (Kamp, 1981)	*Discourse Representation Structure* or *DRS* (a semantic representation of a discourse using a first-order predicate logic)	-	A representation of a discourse is aggregated from sentence-level representations without considered the structure of the discourse.	
Linguistic Discourse Model (LDM) (Polanyi, 1996)	A semantic representation similar to the one used in DRT but with a slightly different representation for discourse context.	-	A *discourse parse tree* (A discourse segment is a semantically motivated unit which is equivalent to a clause while a relation is a syntactic or a semantic connection between the segments)	
Segmented Discourse Representation Theory (SDRT) (Asher, 1993)	A semantic representation similar to the one used in DRT.	-	A *Segmented Discourse Representation Structure* or *SDRS* (A discourse segment is a proposition which is equivalent to a simple DRS in DRT while a relation is a *semantic* or *structural* relation that specifies how the semantic representations of the segments should be combined)	
Rhetorical Structure Theory (RST) (Mann and Thompson, 1988)	*Subject matter* relations such as Elaboration and SolutionHood	*Presentational* relations such as Motivation and Justify	A *rhetorical structure* contains rhetorical relations between utterances and group of utterances (focus on the relations between segments)	The theory focuses more on the informational perspective than the intentional perspective (Moore and Pollack, 1992)

Table 2.1: Discourse structure models comparison

Discourse structure model	Informational perspective	Intentional perspective	Compositional structure	Remarks
Intentional-oriented				
Speech act theory (Searle, 1975)	-	Speech act describes the role of each utterance with respect to a speaker's intention and its effect on a listener	A compositional structure of a dialog is not described. The theory focuses only on the utterance level.	Domain-specific acts can be added.
Dialog Act Markup in Several Layers (DAMSL) (Core and Allen, 1997)	*Utterance Features* (e.g. the content and form of an utterance)	*Forward Communicative Functions* (similar to speech acts)	*Backward Communicative Functions* capture the relations between the current utterance and the previous ones in the form of a link to an antecedent. However, these relations only link two utterances together. Discourse segments and structural relations among the segments are not specified.	The model describes the role of each utterance with multiple labels in multiple layers. More layers that describe other types of utterance functions, such as domain-specific information, can be added
Dialog grammar (Sinclair and Coulthard, 1975)	-	Terminal nodes in the grammar are dialog acts	A hierarchical structure of recurrent patterns in a dialog (focus more on segments than relations)	
Plan-based models (Cohen and Perrault, 1979)	Conditions, constraints and arguments of an action	A plan describes how speaker intentions (speech acts) fit together in a dialog in order to achieve a dialog goal	A compositional structure is not mentioned directly, but a plan can be decomposed into small steps (sub-plans).	

Table 2.1: Discourse structure models comparison (cont.)

Discourse structure model	**Informational perspective**	**Intentional perspective**	**Compositional structure**	**Remarks**
Intentional-oriented (cont.)				
The theory of conversation acts (Traum and Hinkelman, 1992)	-	*Core speech act level*	The level of *argumentation acts* combines core speech acts into a hierarchy of higher level discourse acts. Some argumentation acts resemble the rhetorical relations while some argumentation acts resemble adjacency pairs.	Each level represents different pieces of information. The theory also includes *turn taking acts* and *grounding acts* (emphasizes more on coordinated activities)
Grosz and Sidner's Theory of discourse structure (GST) (Grosz and Sidner, 1986)	Entities in the *attentional state*	*Discourse Segment Purpose* or *DSP*	The *linguistic structure* or the structure of utterances describes coherent segments in a discourse (a discourse segment is defined based on intention) while the *intentional structure* models the relations between the purposes of these segments (between DSPs)	The theory also models the *attentional structure* (the structure of the focus of attention)

Table 2.1: Discourse structure models comparison (cont.)

The second group of discourse structures in Table 2.1, intentional-oriented discourse structures, includes speech act theory, Dialog Act Markup in Several Layers (DAMSL), a dialog grammar, a plan-based model, the theory of conversation acts, and Grosz and Sidner’s Theory of discourse structure (GST). Among these theories and models, speech act theory and its successor, DAMSL, only focus at the level of an individual utterance. Both of them model a speaker’s intention in uttering each utterance without describing how the intentions fit together in a dialog. In DAMSL, richer information about the utterance’s functions is provided. Even though speech act theory does not describe the overall structure of a dialog, a speech act, which captures an utterance-level intention, is a key component in many other theories that use an utterance as a dialog structure unit including a dialog grammar, a plan-based model and the theory of conversation acts. These theories model the structure of a dialog from the intentional perspective by describing how utterance-level intentions captured by speech acts or dialog acts fit together in a dialog. In a dialog grammar, a dialog act is the smallest unit of recurring patterns in a dialog. A plan-based model, on the other hand, uses a plan to describe how speaker intentions captured by speech acts fit together in a conversation and how they relate to the conversation goal. In the theory of conversation acts, the structure of a dialog is accounted for by argumentation acts which combine speech acts into a hierarchy of higher level discourse acts. GST, on the other hand, uses a larger discourse unit, a sequence of utterances. Nevertheless, this discourse segment is also defined based on intention and goal. The structure of a discourse is modeled in terms of the relations between the purposes of the discourse segments

Since the discourse structure representations reviewed in the previous sections capture different aspects of a dialog, they are applied differently in a dialog system. A dialog structure that captures the intentional aspect of a dialog, such as a plan-based model and Grosz and Sidner’s Theory of discourse structure, is employed in a natural language understanding module to help interpreting user utterances. For instance, the TRAINS system (Ferguson et al., 1996) utilizes a plan-based model and a plan recognition algorithm in order to guide the interaction between a user and the system. The current state of the plan and discourse context are used to interpret the underlying intention of a user utterance and determine an appropriate system response. On the other hand, a dialog structure model that captures relations between discourse segments, such as RST, is utilized in a natural language generation module. In the MATCH system (Stent et al., 2004), where an output utterance may contain complex information such as a list of restaurants and a comparison among them, the sentence planner that models the rhetorical relations among various pieces of information produces a higher quality output utterance.

In addition to various dialog structure theories and models discussed in previous sections, *information state theory* (Larsson and Traum, 2000) is another theory that describes the structure of a dialog. However, instead of defining a specific dialog structure representation, information state theory provides a general dialog modeling framework that can be interpreted and implemented in the context of any dialog structure theory. Under this general modeling framework, it is possible to directly compare two dialog structure modeling approaches when they are used to implement the same dialog application. Information state theory centers on the concept of *information state*, a representation which captures relevant information in a dialog that is necessary for distinguishing one dialog from the others. This information also includes the accumulative information from previous actions and the obligation for future actions. The key idea of this theory concerns the representation of the information state, how it is updated and how the updating process is controlled. Information state theory consists of: *informational components* (e.g. domain knowledge, intentions, and a user model), *formal representations* of the informational components, *dialog moves* that trigger the update of the information state, *update rules* which formalize the way that the information state is changed as a dialog progresses, and *update strategy* which selects an appropriate update rule. The term dialog move in information state theory is an abstract term for any mediating input and not restricted to just a speech act. The architecture and tools that facilitate the implementation of the information-state approach is available in TrindiKit, a dialog management toolkit developed under the TRINDI project (Larsson and Traum, 2000). The toolkit has been used to develop many dialog system managers that employ different dialog processing techniques. For example, GoDiS (Kruijff-Korbayová et al., 2003), an information-seeking dialog system in multiple domains, represents the information state as a record while MIDAS (Traum et al., 2000), a dialog system in a route-planning domain, uses Discourse Representation Structure (DRS) as information state representation.

The goal of this dissertation is to develop a machine learning approach that can infer, from a corpus of in-domain conversations, the domain-specific information required to build a task-oriented dialog system; therefore, a suitable dialog structure representation for this purpose needs to capture all of the necessary domain information. This domain-specific information includes a list of tasks that a dialog system has to support, and how a complicated task should be decomposed into a set of sub-tasks. This information also includes domain keywords which capture pieces of information that dialog participants need to communicate in order to achieve each task or sub-task. In a retail domain, for instance, a product name and a quantity are domain keywords since they are essential information for making a purchase.

A hierarchical structure of a task and its sub-tasks can be considered as a compositional structure of a dialog where a discourse unit is defined based on the characteristics of the task. Many of the dialog structures reviewed in the previous sections represent a compositional structure of a dialog. However, most of them define a discourse unit at the level of a sentence or a clause, which is too small to be a step in a task, except for Grosz and Sidner's Theory of discourse structure (GST) which uses a larger discourse unit, a sequence of utterances. For those discourse structures that use a sentence or a clause as a basic unit, some discourse segments at the upper levels of the compositional structure (such as lessons and transactions in Sinclair and Coulthard's (1975) dialog grammar, the components at the top levels of a rhetorical structure, and the argumentation acts at the top levels of the hierarchy in the theory of conversation acts) may resemble tasks and sub-tasks.

A domain keyword, another piece of the required domain-specific information, is an actual content of a dialog that the participants have to communicate in order to accomplish a task; therefore, a suitable dialog structure representation must capture the informational aspect of a dialog. Nevertheless, the informational-oriented discourse structures reviewed in Section 2.1.1, such as the Linguistic Discourse Model (LDM), Segmented Discourse Representation Theory (SDRT), model the meaning of a discourse with a semantic representation instead of the actual entities that were mentioned in the discourse.

There are some intentional-oriented dialog structures that also model the informational aspect of a dialog including Dialog Act Markup in Several Layers (DAMSL), a plan-based model, and Grosz and Sidner's Theory of discourse structure (GST). In the original DAMSL annotation scheme, the Information Level in the Utterance Features layer only captures abstract characteristics of an utterance (e.g. whether the utterance addresses a task, a communication process, or other aspects), but not the actual information that the utterance carries nor the information that is specific to a particular domain (e.g. the type of a task). However, some extensions of the DAMSL annotation scheme do capture the actual content of a dialog and some domain-specific information. In the COCONUT project (Eugenio et al., 1998), the *Topic* layer contains domain-specific tags such as needItem, haveItem budgetAmount, and budgetRemains that describe the content of an utterance. Nevertheless, some of them seem to capture domain-specific intentions rather than domain-specific entities enclosed in the utterance. *ItemFeature* is another set of labels that captures domain-specific information. However, only the properties of domain objects (e.g. price and color) are annotated, not the objects (e.g. table and chair) themselves.

Hardy et al.'s (2003) annotation scheme is another extension of DAMSL that captures domain-specific information. The semantic layer was added to the original DAMSL annotation scheme in order to model domain-related information enclosed in each utterance. The information captured by this layer consists of *transactions* (or *AccessFrames*) which contain *attributes* (or *slots*) and *attribute modifiers*. In a customer service domain, AccessFrames correspond to customer-service tasks such as ChangeAddress; this task contains an attribute Address and its modifier New, for example. While this semantic layer does capture the informational-aspect of a dialog that is also domain-specific, it restricts itself to utterance-level information similar to the DAMSL annotation scheme that it is based on. For instance, an AccessFrame only captures the name of a task that each utterance belongs to rather than the discourse segment that corresponds to the entire task.

In some plan-based models such as (Litman and Allen, 1987) and (Lambert and Carberry, 1991), the parameters of the domain plans are quite similar to the notion of domain keywords, items of information that dialog participants need to communicate in order to achieve a task. Grosz and Sidner's Theory of discourse structure also mentioned objects in a discourse segment when discussing the attentional state but did not provide a detail description about these objects. Information state theory is another theory that includes informational components as one of the elements in its framework. However, since information state theory is a general dialog modeling framework, the choice of the informational components depended on the choice of the dialog structure theory that will be adopted in the framework.

In summary, there are several existing discourse structure models which represent a compositional structure of a dialog that is similar to a hierarchical structure of a task and its sub-tasks. These discourse structure models are Rhetorical Structure Theory (RST), a dialog grammar, the theory of conversation acts, and Grosz and Sidner's Theory of discourse structure (GST). Nevertheless, the entire discourse structure in the first three models (RST, a dialog grammar, and the theory of conversation acts) does not correspond to the task structure as the discourse units that are lower in the discourse structure hierarchy are smaller than steps in a task. The intentional structure in GST is the one that is most similar to the task structure; however, GST does not explicitly model domain keywords.

Only a few existing dialog structure representations capture the informational aspect of a dialog that resembles domain keywords, items of information that dialog participants need to communicate in order to achieve a task. These dialog structure representations include some variations of a plan-based model and the extension of the DAMSL annotation scheme proposed by Hardy et al. (2003). However, Hardy et al.'s annotation scheme only focuses at the level of an individual utterance and does not describe the overall structure of a dialog. Although a plan-based model doesn't address a compositional structure of a dialog directly, it allows a plan to be decomposed into smaller steps. Thus, a plan-based model appears to be a dialog structure representation that captures all of the domain-specific information required to build a dialog system. Nevertheless, one difficulty in modeling the required domain-specific information with a plan-based model is the complexity of the model. In addition to the required domain-specific information, a plan-based model includes many intentional components such as beliefs and intentions (represented by speech acts). These intentional components, while capturing useful information for processing a dialog, do not directly describe the tasks that a dialog system has to support or the domain-specific components required to achieve the tasks. Moreover, intentional components are rather abstract and may be difficult to be identified directly from in-domain conversations through an unsupervised machine learning approach. The reason for using an unsupervised learning approach rather than a supervised one is discussed in Section 2.2.3.

Since none of the existing discourse structure representation is suitable for the purpose of this dissertation, which is to infer the domain-specific information required to build a task-oriented dialog system from in-domain conversations using an unsupervised machine learning approach, a new representation, called a *form-based dialog structure representation*, is proposed. The form-based representation captures all the required domain-specific information and focuses only on concrete information that can be observed directly from in-domain conversations. Chapter 3 describes the proposed form-based dialog structure representation in detail. A comparison between the proposed dialog structure representation and existing discourse structure representations is also discussed.

2.2 Data-driven approaches to dialog structure modeling

In the past decade, the computational linguistics community has focused on developing language processing approaches that can leverage the vast quantities of corpus data that are available. The same idea has also been applied by dialog system researchers and developers. As more dialog data becomes available, techniques for building dialog systems have been shifted from hand-crafted approaches toward data-driven ones. There has been substantial amount of research on applying data-driven approaches to several dialog system components. Those works are different in terms of the algorithm used, the component to be learned, and how the learning approach is integrated with a dialog system. Reinforcement learning is one best-known approach for learning a dialog management policy from in-domain dialogs. Singh et al. (2002) is among the first groups who successfully applied this technique to find an optimized policy from very large policy space as demonstrated in the NJFun system. Karahan et al. (2003), who combined two classifiers (the Bayesian classifier and Boosting) in order to identify users' intents (i.e. call-types) in a customer care application, are among many other researchers that applied a machine learning technique to the problem of natural language understanding in a goal-oriented spoken dialog system. Various approaches to this problem are summarized in (Bangalore, Hakkani-Tür et al., 2006). As for natural language generation, Stent et al. (2004) trained a sentence ranker to select an appropriate sentence plan from a set of possible ones by applying a boosting algorithm on human-rated sentences.

Research on a data-driven approach to dialog structure modeling is relatively new and focuses mainly on recognizing a structure of a dialog as it progresses. Since a dialog structure encapsulates relations between utterances and dialog context (e.g. between user intentions and a task being pursued), a dialog system can utilize this information to better understand a user's utterance and generate an appropriate response to the user. Various dialog structure recognition approaches will be discussed in more detail in Section 2.2.1.

A data-driven approach to dialog structure modeling can also be used to reduce the amount of human effort spent in the knowledge engineering process when developing a dialog system in a new task-oriented domain. Necessary knowledge required to build a dialog system could be identified through a machine learning approach rather than hand-crafted. For example, a task model can be learned with an example-based learning algorithm as described in (Garland et al., 2001) and in Section 2.2.2.2. Acquiring the necessary knowledge from data is an acquisition process that is carried out before a dialog system is created. This is contrasted with a dialog structure recognition process discussed previously where pre-specified dialog structure components are recognized as a dialog progresses. Data-driven approaches for a dialog structure acquisition problem have only been explored by a handful of researchers. Some of the interesting works in this area are reviewed Section 2.2.2.

Research works on dialog structure learning (both recognition and acquisition) differ from each other in two important aspects: the type of the structure to be learned and the learning approach. A variety of dialog structure theories are adopted in dialog system implementation and in some cases they are modified to better suit the tasks. The choice of the learning approach depends heavily on the characteristics of the dialog structure and the type of information available for training. For example, a Markov model is suitable for sequential structures while a grammar induction approach is suitable for hierarchical structures. For a multi-level dialog structure and a hierarchical structure, the components of the structure are often identified independently or in a cascaded manner where information from one component is being used to identify another component. This decomposition helps reduce learning complexity. Related works reviewed in the following sections are organized according to the learning approaches. The overview of the proposed learning approach for acquiring the form-based dialog structure representation is given in Section 2.2.3.

2.2.1 Dialog structure recognition approaches

2.2.1.1 Markov models

A Markov model is suitable for learning the sequential structure of observations. Since some dialog structure components, such as dialog acts, seem to have a sequential property, the Markov model has been widely used in many dialog structure learning approaches. In (Woszczyna and Waibel, 1994), the structure of a conversation in a scheduling domain composes of topics, discourse states, speech acts, and common phrases. The information captured by the dialog structure can reduce ambiguities in natural language understanding. Two components of the structure, dialog state and speech act, were focused in the paper. To infer both dialog structure components automatically from data, a Markov Model (MM) was used in a supervised scenario and a Hidden Markov Model (HMM) was used in an unsupervised scenario. Only word sequences were used as features in both models. The Markov model requires training data annotated with state labels, which in this case are equivalent to dialog states and speech acts. The Hidden Markov model, on the other hand, requires no labeled data; therefore, it can utilize all of the data available. The notion of state is obtained automatically from the data given a number of hidden states. The Hidden Markov model performed better than the Markov model in terms of perplexity when both models were trained on the same amount of data as a pre-defined set of states could be suboptimal for a given set of data. The Hidden Markov model could be improved further by adding more training data and increasing the number of hidden states; however, the latter came with higher computational cost.

Finke (1998) used a Markov model to both segment and classify speech acts in telephone-based conversations in the CALLHOME SPANISH corpus. Speech acts are part of the three-level discourse structure consists of speech acts, dialog games (similar to the ones described in Section 2.1.2.3), and discourse segments or topic segments. The structure was developed under the CLARITY project (Levin et al., 1998) which aimed at exploring the use of discourse structure in dialog understanding. The annotation scheme for speech acts was extended from the DAMSL annotation scheme in order to handle non-task-oriented conversations in this domain. On a speech act segmentation problem, a Markov model was trained on word and part of speech features, and achieved a comparable performance to a neural network approach. To classify the speech act of each segment, a Markov model was trained on prosodic features, word sequences, and speech act sequences. The integrated Markov model for both segmentation and classification was also investigated. For a topic segmentation problem, Hearst's TextTiling algorithm (Hearst, 1997) was used to determine topical segment boundaries.

2.2.1.2 Grammar induction approaches

A grammar induction approach can be used to identify the structure of a dialog if the structure can be described by a context-free grammar. The VERBMOBIL system used a plan hierarchy to describe a dialog in a meeting scheduling domain (Bub and Schwinn, 1996). A plan hierarchy is a four-level organization composes of the dialog act level, the turn level, the phrase level, and the dialog level. By viewing plan recognition as parsing, the plan hierarchy is compiled into a context-free grammar. Grammar rules (or plan operators) for processing the components in the dialog act level, the phrase level, and the dialog level can be hand-coded. However, as the number of turn classes is quite large and the sequences of dialog acts that correspond to each turn class are rather complex, it is difficult to construct plan operators that generalize for all of the data by hand. Alexandersson and Reithinger (1997) used a grammar induction approach based on Bayesian model merging to derive a stochastic context free grammar that describes the structure of each turn class from the corpus of dialog act annotation. The automatically derived plan operators were applied in a plan recognition process to identify the intentional structure of user utterances; 66.8% turn class prediction accuracy was reported. They also suggested the focus and relations between new utterances, and the current foci as additional information sources for improving the performance.

In recent research, Bangalore, Fabbrizio et al. (2006) attempted to recognize a structure of a task-oriented dialog as it progresses in order to guide a dialog manager's decision and construct an appropriate agent response. Other dialog system components besides the dialog manager and the natural language generation module could also benefit from the information captured by a dialog structure as well. Based on the SharedPlans theory (Grosz and Sidner, 1990) adopted in this research, a dialog structure was represented as a tree that encapsulates the task structure, the dialog act structure, and the linguistic structure of utterances, which contains the inter-clausal relations and predicate argument relations within a clause. The paper focused on recognizing the task structure of an on-going dialog in a catalog ordering service domain. A top-down incremental parser that incorporates bottom up information was used to discover the most likely plan tree that encapsulates the dominance relations (or hierarchical relations) between sub-tasks from a sequence of utterances. The utterances were first segmented and classified into a sequence of sub-tasks with a maximum entropy classifier in order to identify the precedence relations (or sequential relations) between sub-tasks. For each utterance, the classifier predicted the most likely sub-task label given word n-gram features of local context. Since the label not only represents the type of sub-task but also encodes the position of the utterance in relative to the sub-task (i.e. begin, middle and end), the classifier can segment a dialog into a sequence of sub-tasks and assign a label to each sub-task in a single parse. The paper also discussed dialog structure recognition at the level of dialog acts.

2.2.1.3 Categorical classifiers

Another type of machine learning algorithm that has been extensively used for dialog structure recognition when the sequential structure of the components is not fundamental is a categorical classifier; a neural network and a decision tree, for example. Vilar et al. (Vilar et al., 2003) used both a neural network and a Hidden Markov Model to identify the structures of dialogs in the Spanish train information domain. The structure consists of speech acts, *frames* and *cases*. Each frame represents a specific type of user message and contains a set of cases or slots that associate with pieces of information that are related to a query. Both frames and slots are domain-specific components and could be used to improve the understanding process of the system. The authors of this paper assumed that while the sequential structure of a sentence is useful for segmenting the sentence into a set of slots, the sequential structure is not fundamental for classifying the type of frame. For that reason, a neural network was used to classify the frame type of each user turn given context word features while a frame-specific HMM trained on annotated data was used to segment each turn into a sequence of semantic units. Each semantic unit captures the semantic function of a word or a group of words and corresponds to a HMM state. Particular types of semantic units are associated with the slots. The proposed techniques achieved 5.2% error rate on frame classification and 14.4% on semantic unit segmentation.

In (Hardy et al., 2004), a vector-based approach was used to train both a task identification agent and a dialog act classifier in order to identify the customer's desired *transaction* and the corresponding dialog act of each utterance respectively in the user-initiative customer service system, Amities. Both a transaction (also *task* and *frame*) and a dialog act are components of the dialog structure proposed by Hardy et al. (2003) discussed in Section 2.1.3. The vector-based approach used a cosine similarity score to determine the similarity between the vector that represents an input utterance and the vector that represents each task or each dialog act created from training data.

Speech acts or dialog acts may be used independently in a dialog system without specifying the structure of an entire dialog. Many classification algorithms, such as a decision tree and a maximum entropy model, have been used to predict a dialog act label of a give dialog segment such as an utterance. Related works on dialog act classification were summarized in (Stolcke et al., 2000). However, the research in this area is less relevant to the work in this dissertation which focuses more on identifying the overall structure of a dialog.

Categorical classifiers require a set of pre-defined categories and, for each category, the training data. However, both requirements are not applicable when acquiring domain-specific information in a new domain, as in the case of this dissertation research, since the target representations will be explored from in-domain dialogs instead of being pre-specified.

2.2.2 Dialog structure acquisition approaches

2.2.2.1 Conceptual clustering

Möller (1998) developed a dialog modeling toolkit, DIA-MOLE, to help reduce human effort in creating a dialog model for a new application. Instead of using a pre-defined dialog act taxonomy, an unsupervised learning technique was used to infer a set of *domain-specific dialog acts* (*DDAs*) from a corpus of in-domain conversations. The

DDA learner module in DIA-MOLE utilizes a conceptual clustering algorithm, CLASSITALL, to create a DDA hierarchy from segmented utterances and their features. A set of features for each segment, which consists of prosodic events, recognized words, and semantic structure, is extracted from various knowledge sources available to a dialog system and is represented by a set of attribute-value pairs. CLASSITALL allows various types of features including numeric, symbolic, and structured features to be integrated into the clustering framework. Moreover, each feature can be associated with a probability value, which expresses the quality of the feature (e.g. a confidence score produced by a feature extraction algorithm), or a weight, which specifies the significance of the feature.

Since there is no example from training data to supervise the clustering algorithm, CLASSITALL uses a heuristic that reflects the quality of the clusters to guide the hierarchy construction. Based on the assumption that a good set of clusters is the one that similar objects are assigned to the same class while dissimilar objects are assigned to different classes, CLASSITALL defines a cluster quality measure, *category utility*, as a tradeoff between intra-class similarity and inter-class dissimilarity. Both intra-class similarity and inter-class dissimilarity are computed from a conditional probability of an attribute-value pair and a class.

DIA-MOLE was applied in VERBMOBIL, a dialog system in an appointment scheduling domain, where predicted DDAs could help identify an appropriate language model for a speech recognizer or guide a spoken language generation module. The learned DAA taxonomy was evaluated against the human-assigned taxonomy; comparable dialog act prediction rates were reported.

2.2.2.2 Example-based learning

Garland et al. (2001) used an example-based learning algorithm to lessen domain expert effort in developing a *task model*, a declarative representation of a task, for a collaborative system. The task model, which based on the SharedPlans theory of collaborative discourse (Grosz and Sidner, 1990), captures the structure of actions. The task model composes of *actions* and *recipes*. There are two types of actions, a primitive action, which can be executed directly, and a non-primitive action, which can be achieved indirectly by achieving other actions. A recipe describes a set of steps required in order to achieve a goal or a sub-goal (a non-primitive action). It also contains constrains on the order of actions and the logical relations among action parameters. A collaborative system, which helps a user achieves a task goal through a spoken conversation, requires domain-specific task models in order to adapt agent utterances according to the task.

With an example-based learning approach, a domain expert only needs to generate examples of how to accomplish a task which is considered more intuitive than constructing a complete task model that generalizes for all possible cases. The learning algorithm then infers the target task model by inducing the constraints and generalizing the model over a series of annotated examples. The approach was tested on two simulated tasks: building graphical user interfaces and cooking, but not on real dialogs. Expert examples, which were described in terms of actions and relations among them, were generated from the targeted task model. The numbers of examples required to learn the correct task models given different kinds of annotations were reported.

2.2.2.3 Information extraction

Feng et al. (2003) proposed a framework called WebTalk that aimed at creating a specific type of a dialog system, namely a customer care service, automatically from information extracted from the company's website. There are three types of customer care dialog systems: 1) an information retrieval aid system (or a question-answering system), 2) a form-filling system, which helps a customer fills out an online form, and 3) a table-based system, which operates on a table of related information (e.g. product details) automatically created from web contents. A corporate website contains rich and well-organized information including a web page structure, hyperlinks between related web pages, lists, forms, tables, and graphics; hence, it is a useful resource for extracting task-specific information. The task-specific knowledge includes a website structure and an *information unit*, a coherent area in a web page according to its content or its behaviors such as LIST-ITEMS and QUESTION-ANSWER. A website structure is obtained from directory organization and a list of hyperlinks. Information units are generated by Webpage Parser which identifies the boundaries and type of each information unit using a supervised classifier, a support vector machine (SVM). Some types of information units may need further processing to extract more useful information. Since the information source is a company's website rather than a corpus of dialogs, the information extracted is not the structure of a dialog or a component in the structure. Nevertheless, those extracted pieces of information are included in a task-specific knowledgebase that will be used by a customer care service system

2.2.3 The overview of the proposed learning approach

The goal of this dissertation is to infer the domain-specific information required to build a task-oriented dialog system from in-domain conversations through a machine learning approach. Acquiring the necessary domain knowledge from a set of human-

human dialogs is considered a knowledge acquisition process and is carried out before a dialog system is created. This is contrasted with a dialog structure recognition process in which pre-specified dialog structure components are recognized as a dialog progresses.

Most data-driven approaches to dialog structure recognition discussed in Section 2.2.1 rely on supervised learning algorithms since they usually provide more accurate results but at the cost of manually labeled data. However, for the dialog structure acquisition problem investigated in this thesis, the structure of a dialog in a new task-oriented domain has not been pre-specified and will be explored from data. Hence, a corpus of in-domain dialogs annotated with the target dialog structure is not available for training a supervised learning algorithm. The example-based learning algorithm and the information extraction algorithm discussed in Section 2.2.2.2 and 2.2.2.3 had to utilize annotated data from other information sources, examples described in a specific annotation language and a well-organized website, respectively. In this thesis an unsupervised learning approach is preferred since the goal is to minimize human effort including annotation effort in the domain knowledge engineering process. Since this process occurs before the first prototype system is created, a corpus of recorded conversations between the humans who perform the same task as the target dialog system becomes the main resource. The motivation behind the use of an unsupervised learning approach is quite similar to the idea that motivates the work in DIA-MOLE, the only approach among the dialog structure acquisition approaches discussed in Section 2.2.2 that utilizes an unsupervised learning algorithm. Nevertheless, different unsupervised learning algorithms that are suitable for the target dialog structure, the form-based dialog structure representation, are investigated in this thesis. The detail discussions of those algorithms are provided in subsequent chapters.

To make the problem tractable, I divide a dialog structure acquisition problem into two sub-problems: concept identification and clustering, and form identification (a form is associated with a sub-task in the form-based dialog structure representation). Each sub-problem is handled separately and is discussed in more detail in Chapter 5 and Chapter 6 respectively. However, it should be kept in mind that these individual components are parts of the same dialog structure; therefore, information about one component may be useful for inferring another component. After each component can be acquired with acceptable accuracy, interaction between components should also be considered in the learning process. The decomposition of a dialog structure learning problem for a multi-level dialog structure and a hierarchical dialog structure was also applied by many researchers (Finke et al., 1998; Hardy et al., 2004; Vilar et al., 2003).

Chapter 3

Form-based Dialog Structure Representation

The goal of this dissertation research is to develop a data-driven approach that can infer domain-specific information required to build a task-oriented dialog system from a corpus of in-domain conversations. In order to achieve this goal, one would have to first specify a suitable domain-specific information representation, and then develop a machine learning approach that is able to identify the domain information captured by this representation from a corpus of in-domain dialogs. This chapter focuses on the first step, describing a suitable domain-specific information representation and demonstrating how it can be used to model domain information in various types of task-oriented dialogs.

A domain-specific information representation that is suitable for the purpose of this dissertation should have all of the following properties: sufficiency, generality, and learnability.

- *Sufficiency* is implied if the representation captures all domain-specific information required to build a task-oriented dialog system.

  This domain-specific information includes a list of tasks that a dialog system has to support, for a complicated task how it should be decomposed into smaller steps or sub-tasks, and domain keywords which capture pieces of information that dialog participants need to communicate in order to achieve each task or sub-task. For instance, in a retail domain, where a task is to make a purchase, the domain keywords are a product name and a quantity. These two pieces of information are essential for making a purchase.

- *Generality* is implied if the representation can describe task-oriented dialogs in dissimilar domains and types.

  Different types of task-oriented domains have different characteristics; for instance, some discourse phenomena such as grounding do not occur in every domain. Therefore, a desired domain-specific information representation should be generalized for various types of task-oriented domains, namely, it should be domain-independent.

- *Learnability* is implied if the representation can be identified by a machine learning algorithm from observable language behaviors in human-human conversations.

  When acquiring the domain-specific information for a new task-oriented domain, there is no annotated data available for training a supervised learning algorithm as the target domain-specific information has not been specified and will be explored from in-domain conversations. Since we have to rely on an unsupervised learning approach, the representation of the domain-specific information has to be observable from the conversations.

Existing dialog structure representations do not have all of the three required properties. Most of them do not capture all of the domain-specific information required to build a dialog system as discussed in Section 2.1.3. Some of these discourse structure models and theories describe the compositional structure of a dialog that resembles tasks and sub-tasks but do not represent domain keywords, or vice versa. For the one that captures all of the required domain-specific information, namely, a plan-based model, its discourse structure is quite complex and contains many intentional components, such as beliefs and intentions. These abstract components are rather difficult to be observed directly from a conversation and, as for the current technology, may not be learnable through an unsupervised machine learning approach. Moreover, these additional components, while capturing useful information for processing a dialog, do not directly describe a task or domain-specific elements required to achieve the task.

In order to have a domain-specific information representation that has all of the desired properties discussed above, we have to either augment an existing dialog structure representation or specify a new representation. In this thesis, I propose a new representation, called a *form-based dialog structure representation*, as a target representation of the domain-specific information that will be inferred from in-domain conversations. This representation is based on the notion of *form*, a data representation used in the form-based dialog system architecture. The form-based dialog system architecture is described in detail in section 1.1.2. I choose to develop a new dialog structure representation based on the data representation used in a functional dialog system architecture rather than augmenting an existing dialog structure theory because this data representation already captures the domain-specific information a dialog system needs to have in order to support a task. It is quite easy to demonstrate that the representation is sufficient for representing task-oriented conversations which is one of the three required properties. The *sufficiency* of the form-based dialog structure representation has been demonstrated by the success of the systems that were implemented based on the form-based dialog system architecture. Another advantage of using an existing dialog system framework to describe the structure of a task-oriented conversation is that the connections between the dialog structure components and the components of a dialog system that employs the structure become straightforward. This direction is opposite to many other approaches that implement a dialog system from an existing dialog structure theory.

In addition to *sufficiency*, the form-based dialog structure representation needs to have other two properties: *generality* and *learnability*. The form-based dialog system architecture has been used mainly in information-accessing domains where a form corresponds to a database query form while slots in the form represent search criteria. In this thesis, a more *generalized* definition of the form representation is provided, so that it can be used to represent the structure of dialogs in other types of task-oriented domains as well. In terms of *learnability*, the form-based dialog structure representation focuses only on concrete information that can be observed directly from in-domain conversations; hence, it should be *learnable* through an unsupervised learning approach. In the following parts of this section, I will describe the form-based dialog structure representation and discuss its properties in more detail. The approaches that are used to verify these properties are also described.

The form-based dialog structure representation is a three-level structure of task, sub-task, and concept. This representation models the *tasks* that a dialog system has to support, a set of *sub-tasks* (a decomposition of a task) which corresponds to the steps that needs to be taken in order to successfully accomplish the task, and *concepts* which are the items of information (or domain keywords) that dialog participants have to communicate in order to achieve a task or a sub-task. The components of the form-based dialog structure representation (i.e. task, sub-task, and concept) reflect the observable structure of a task-oriented conversation discussed in Section 1.2. A hierarchical structure of a task and its sub-tasks represents a compositional structure of a dialog that is defined based on the characteristics of the task while a domain concept captures the actual content being conveyed in the dialog. Along the two aspects of a discourse (*informational* and *intentional*) discussed in Section 2.1, the form-based representation focuses more on the informational aspect as it represents the actual content of the discourse. The intentional aspect is addressed by a conversation goal which is included in the definition of a task. A formal definition of each component is provided in Section 3.1. As the name indicates, the form-based dialog structure uses a form as a central representation. The concepts that the participants have to communicate in order to achieve a particular sub-task are stored together in the same form. Therefore, a task is represented by one or more forms depending on the number of its sub-tasks.

The use of forms and a form-filling strategy in dialog systems was first introduced by Ferrieux and Sadek (1994) and has been adopted in many systems that built on the form-based dialog system architecture. In the form-based dialog system, a form specifies all relevant pieces of information (or *slots*) that must be filled in before a system can take an action, such as query a database. All domain-specific information a dialog system needs to have in order to support a task is captured by forms; hence, the form-based dialog structure representation is *sufficient* for representing a task-oriented conversation as demonstrated by the success of the systems that were implemented based on the form-based architecture. Examples of these systems are the Philips train timetable information system (Aust et al., 1995) and the CMU Communicator system (Rudnicky et al., 1999). *Dialog coverage*, which measures the percentage of dialog content that can be accounted for by the proposed dialog structure, is also used to verify the *sufficiency* of the form-based dialog structure representation. Dialog coverage is reported in Section 4.1.

The form-based dialog system architecture has been used mainly in information-accessing domains where a form corresponds to a database query form while slots in the form represent search criteria. To make the form representation *generalized* for other types of task-oriented domains as well, a broader definition of the form representation is provided. In this thesis, the notion of form is generalized as a repository of related pieces of information. These pieces of information are not restricted to only the search criteria so that the form can be applied to various types of task-oriented domains. To verify the *generality* of the form-based dialog structure representation, six dissimilar task-oriented domains are analyzed. These six domains are air travel planning (information-accessing task), bus schedule inquiry (information-accessing task), map reading (problem-solving task), UAV flight simulation (command-and-control task), meeting and tutoring. Dialog structure analyses of these six domains are given in Section 3.2 - Section 3.7 respectively. These disparate domains are chosen to cover various types of task-oriented conversations. The choices of domains are also subjected to the availability of human-human data. The corpora of human-human conversations used in the dialog structure analyses are taken from various projects conducted by different research institutes. Some of the corpora were collected during the development process of a spoken dialog system. However, some of the corpora were originally collected for other purposes.

In terms of *learnability*, a dialog structure component that is observable from a conversation should be more easily identified by an unsupervised learning algorithm than a dialog structure component that cannot be directly observed such as a belief and an intention. The components of the form-based dialog structure representation (i.e. task, sub-task, and concept) can be observed directly from a dialog as it reflects the observable structure of a task-oriented conversation discussed in Section 1.2. Task and sub-task represent the decomposition of a complicated task while concept is an item of information that dialog participants have to communicate in order to achieve the conversation goal. By focusing only on the observable structure of a dialog, the form-based dialog structure model works well when all of the domain-specific information necessary for supporting a task is communicated clearly in a dialog. This occurs when a dialog has the following characteristics: 1) the conversation goal is achieved through the execution of domain actions, and 2) the dialog participants have to communicate the information required to perform these actions through dialog. However, if the goal of a dialog is achieved in a different manner or if the necessary domain-specific information is not communicated through the dialog, we may not be able to represent this dialog with the form-based dialog structure representation. The difficulties in representing various types of task-oriented dialogs with the form-based dialog structure representation are discussed in Section 3.8. For the type of dialog that the dialog goal is not directly reflected in the conversation, a more complex dialog structure which also models unobservable aspects of a dialog, such as participants beliefs and intentions, may be required.

Another characteristic of the form-based dialog structure representation that makes it possible to be inferred from in-domain conversations through an unsupervised learning approach is its simplicity. Compared to other dialog structure representations used in other types of dialog system architectures such as a plan-based system, the form-based representation is quite a bit simpler. The detailed discussion about different types of dialog system architectures and the comparison among them can be found in Section 1.1. However, by choosing the representation that is quite simple, we may not be able to model a complex task that has a dynamic structure such as a planning task as discussed in Section 1.1. Nonetheless, the form-based system has been applied successfully in many real world applications. Example of these dialog systems are the Philips train timetable information system (Aust et al., 1995), the CMU Communicator system (Rudnicky et al., 1999), and many other systems built under the RavenClaw framework (Bohus and Rudnicky, 2003).

The *learnability* of the form-based dialog structure representation is verified with the accuracy of the domain information obtained from the proposed machine learning algorithms described in Chapter 5 and Chapter 6. Annotation scheme reliability, which is obtained from a human annotation experiment described in Section 4.2, can also verifies the *learnability* of the form-based representation in terms of human learnability. High annotation scheme reliability suggests that the annotation scheme is concrete and unambiguous which imply learnability.

The rest of this chapter is organized as follows: Section 3.1 provides a detailed description of the form-based dialog structure representation along with the definition of each component. A comparison between the form-based representation and existing dialog structure representations is discussed at the end of Section 3.1. Examples on how to model the structure of a dialog with the proposed representation in six task-oriented domains (air travel planning, bus schedule inquiry, map reading, UAV flight simulation, meeting and tutoring) are given in Section 3.2 - Section 3.7 respectively. These dialog structure analyses are done manually by the developer of the form-based dialog structure representation. The difficulties in applying the form-based dialog structure representation to these task-oriented domains are discussed in Section 3.8. Finally, Section 3.9 summarizes the properties of the proposed dialog structure representation.

3.1 Components in form-based dialog structure representation

In task-oriented domains, participants engage in a conversation in order to achieve a specific goal such as to obtain the departure time of a particular bus or to order a product from a catalog. For simplicity, I will refer to the participants as a client and an operator. Typically the client’s goal is to have the operator perform actions that serve his or her need. The operator in turn needs specific information from the client in order to perform each action. Actions are domain-specific; for instance, in a bus schedule inquiry domain an action is looking up information from a bus schedule while in a retail domain an action is ordering a product. The action occurs when all necessary information has been gathered. For example, in the retail domain, the name of a product and a quantity need to be specified before an order can be placed. The purpose of the goal-oriented conversation is to communicate this information among the participants and to ensure that the information is consistent.

Different task-oriented domains may have some dissimilar characteristics; for instance, some discourse phenomena such as grounding do not occur in every domain. Therefore, in order to develop a dialog structure representation that is generalized across these differences, dialogs from different types of task-oriented domains have to be analyzed. The initial form-based dialog structure representation was derived from the analysis of conversations in information-accessing domains which are the most common application domains of form-based dialog systems. Constraints from a backend database make it easier to identify the form in this type of task-oriented domain. Then conversations from other types of task-oriented domains, namely a problem-solving task and a command-and-control task, were analyzed to verify the completeness of the initial dialog structure. The form-based dialog structure was modified when necessary to account for new discourse phenomena in the new domain. Finally, the definitions of all of the components in the form-based dialog structure representation were verified through several iterations of pilot annotation experiments (human annotation experiments are described in Section 4.2).

In summary, conversations in various task-oriented domains have the following characteristics. In order to achieve a conversation goal, one or more actions must be taken, and all of the information required in order to perform these actions has to be clearly communicated. The form-based dialog structure representation organizes domain-specific information necessary for achieving the conversation goal into a three-level structure of task, sub-task and concept. The definition of each component is given below.

3.1.1 Component definitions and representations

1. A *task* is a subset of a dialog that has one specific goal.

   A simple dialog usually has only one goal; therefore, the entire dialog corresponds to a single task. A complex dialog can have multiple goals. For instance, a customer who makes a call to a customer service may have two goals, to obtain account balance and to change the address; therefore, this dialog consists of two tasks, one for each goal. To decompose a dialog into multiple tasks, each sub-dialog, which corresponds to a task, must have a clear goal that is distinct from the rest of the dialog and can be considered as a separate dialog.

   To accomplish a task goal, one or more actions need to be taken. When multiple actions are required, the task is decomposed into a set of sub-tasks, one for each action. However, if only one action is required, no further decomposition is necessary.

2. A *sub-task* is a step in a task that contains sufficient information to execute a domain action.

Within each sub-task, dialog participants exchange information in order to execute the corresponding action. The sub-task ends when the action is executed. When there is a discussion about the outcome of the action, such as a discussion about the information retrieved from a database, the sub-tasks ends at the end of the discussion

A task can be decomposed into both a sequence of different types of sub-tasks and a series of the same type of sub-task. For example, to reserve a round trip ticket, a client must provide the criteria for a flight in each leg; therefore, a task of reserving a round trip ticket can be decomposed into two sub-tasks: specifying a departing flight and specifying a return flight. In some cases, a sub-task can be further decomposed if it is associated with a complex action. This creates a hierarchical structure of a task and sub-tasks. More examples of the task structure decomposition are given in the following sections.

3. A *concept* is a word or a group of words which captures a piece of information that is necessary for performing an action.

   A piece of information that describes the outcome of an action is also considered a concept.

   Some pieces of information might be complex and contain several components. For example, an address is composed of a street, a city, a zip code, etc. **Street**, **City** or **ZipCode** can be a concept by itself since it captures a distinguishable piece of information that may be used separately. A concept that contains other concepts such as **Address** is called a *structured concept*.

   It is possible that the same word or group of words belongs to more than one concept. For example, "tom@cmu.edu" can be both a **SenderEmail** and a **RecipientEmail**. It is important to distinguish between similar concepts that have different functionalities such as between a sender and a recipient as shown in this example.

Each dialog structure component has two aspects: type and instance. A *type* is an abstraction of similar information items while an *instance* is a specific value of an information item. For example, **query_departure_time** is a type of task in a bus schedule inquiry domain while the dialogs that correspond to this task are instances. For a concept, an instance is also called a *concept member* or a *slot value*. For example, **Color** is a concept type while "red," "blue," and "green" are concept members.

A dialog structure needs to be generalized over all relevant dialogs in a domain. Hence, for the same type of task, some sub-tasks may be optional. Similarly, some concepts may not be required in order to perform the same action. For instance, since not all of the criteria have to be specified in order to retrieve flight information from a database, some concepts, such as **Airline** and **NoOfStop**, are optional.

As the name indicates, the form-based dialog structure uses a form as a central representation. Related pieces of information necessary for performing a particular action, i.e. concepts, are organized into a form. A dialog structure is associated with a form in the following ways. A simple task, which contains only one action, is represented by a single form while a complex task is presented by a set of forms; each of its sub-tasks is associated with one form. We can also say that each form represents information in a subset of a dialog that contributes toward one action. Lastly, a concept is a slot inside a form. Even though a structured concept is composed of a set of concepts, it is not equivalent to a form because there is no action associated with it. A diagram in Figure 3.1 shows how a form captures information from a conversation in a retail domain along with the action that makes use of the information in the form.

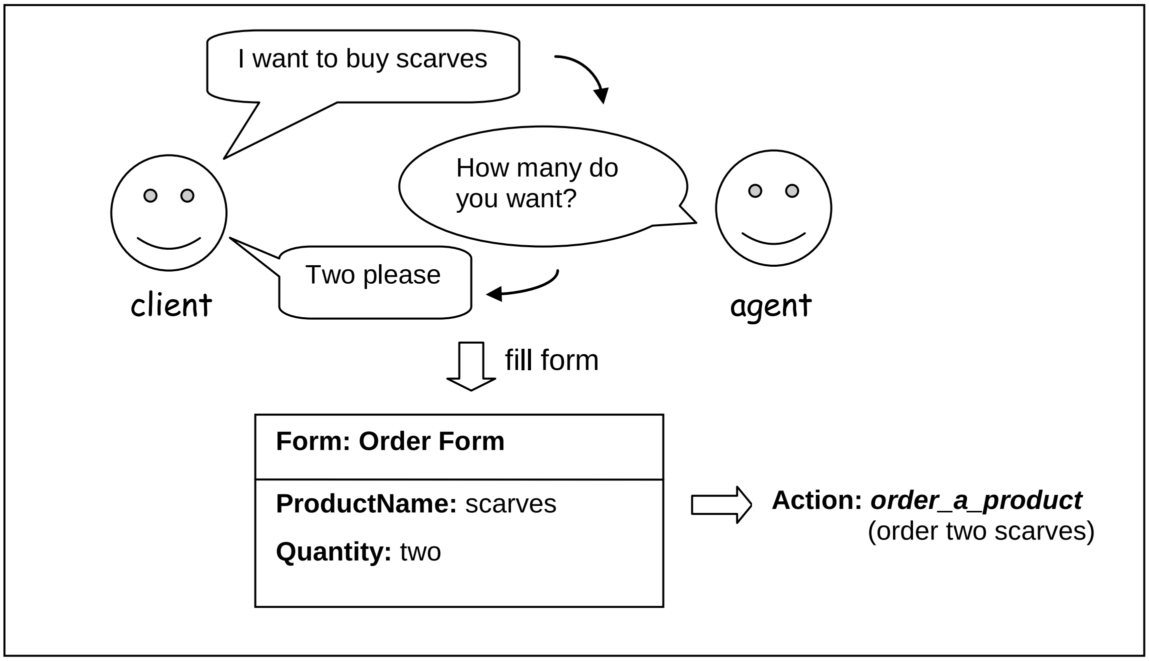


Figure 3.1: A form representation and its associated action in a retail domain

As a conversation progresses, the participants gradually fill in a form with pieces of information. They may also verify the correctness of the information as they try to fill it into the form. When the form is complete (when all of the required pieces of information are obtained) an action that associates with the form is ready to be executed. The purpose of a goal-directed conversation is to fill one or more forms and to ensure that the information is consistent. How each utterance affects the form and its content is described in the next section.

3.1.2 Form operators

In the form-based dialog structure framework, when a participant speaks, his/her utterance is considered as an *operator* (or an *operation*) that operates on a form and its content. At the beginning of a dialog, an utterance fills the corresponding slot in the form with the concept enclosed in the utterance. When the form is complete and its associated action is ready to be taken, an utterance executes the action that is associated with the form or indicates that the action has been executed. After that, the subsequent utterances discuss the outcome of the action. These three basic operations are fill_form, execute_form, and report_outcome, and are summarized in Table 3.1. If the participants are not satisfied with the outcome of the action, they may fill the form with different slot values and then re-execute the action. For example, if a client does not like the flights that were retrieved, he/she may change the search criteria and then ask an agent to retrieve a new set of flights. The new set of slot values get filled into the form through the same fill_form operation as the previous set of slot values.

Each type of operator is defined based on the effect that the operator has on a form and its content, and on the way that the operator uses the information stored in the form. Table 3.1 contains, for each operator, a short description that describes the effect of the operator on a form along with example expressions that indicate this operator. These examples are taken from various task-oriented domains as indicated in the last column of the tables. Air Travel is the Air travel planning domain described in Section 3.2; Bus Schedule is the Bus schedule inquiry domain described in Section 3.3; Map Reading is the Map reading domain described in Section 3.4; UAV is the UAV flight simulation domain described in Section 3.5.

Besides the three basic operations, a dialog participant can also cancel an operation that another participant performs if he/she believes that the operation is not appropriate at that point of the dialog. An example of a cancel operation illustrated in Table 3.1 is taken from the air travel planning domain when an agent canceled a client's previous operation, a fill_form operation which operated on the form of the second leg, in order to continue with the form of the first leg (the current sub-task). If a serious misunderstanding occurs the participant can use a start_over operator to clear the content of the current form and restart the sub-task all over again.

Nevertheless, some utterances may not directly manipulate the content of the form, but rather manage the flow of the communication and maintain the integrity of the dialog. For example, a request_repetition operator repairs a communication problem by requesting another dialog participant to repeat the previous utterance again. This operator doesn't modify or use the content of the form. This group of operators can be regarded as a *discourse-oriented operator* while the first group of operators that directly manipulates the form and is listed in Table 3.1 is considered a *task-oriented operator*. Discourse-oriented operators are listed in Table 3.2.

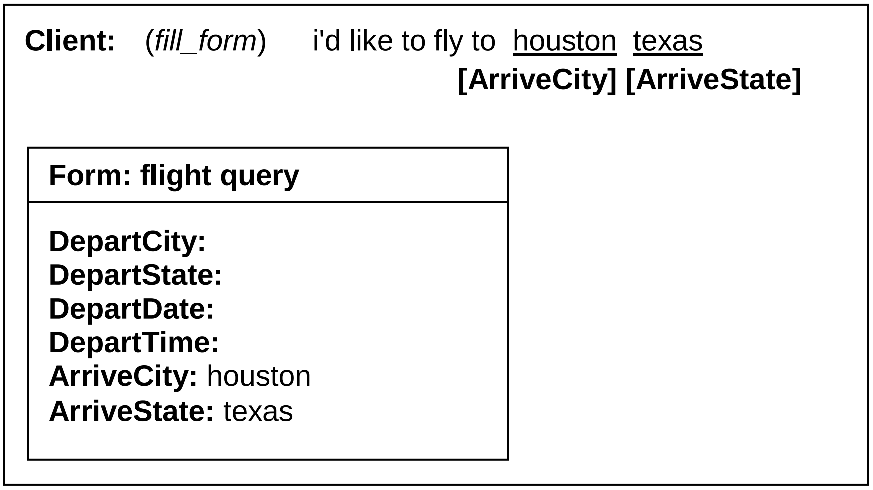
Since the form-based dialog structure model uses the same form representation to represent domain-specific information in every task-oriented domain, the set of operators that can be used to manipulate the form representation is the same across all of the domains. Thus, the lists of form operators in Table 3.1 and Table 3.2 are domain-independent. The consequence of the same operator is the same regardless of the domain. For example, a fill_form operator, which fills a specific slot in a form with a given concept value, has the same behavior in every domain; only the parameters of the operator (the identities of the slot and the form, and the concept value) that are different. The effect of the fill_form operator on the corresponding form in the air travel planning domain and in the map reading domain are shown in Figure 3.2 and Figure 3.3 respectively. However, the consequence of an execute_form operator is the only exception. Since the execute_form operator executes the domain-specific action that is associated with a form, its consequence is domain-dependent. Even though a list of form operators and their effects on a form are domain-independent, the expressions that are associated with each operator vary according to the characteristic of a task as illustrated by example expressions taken from dissimilar task-oriented domains in Table 3.1 and Table 3.2.

Operator	Description	Example utterance	Domain
initiate_form	Initiate a new form (and a new sub-task)	"I also need a car"	Air Travel
fill_form	Fill a slot in a form with a specific concept value	"I'd like to fly to Houston Texas"	Air Travel
		"I'm looking for a 41E leaving downtown Pittsburgh around three o'clock"	Bus Schedule
		"To the left for about an inch"	Map Reading
		"AVO, radius of H-area is five miles"	UAV
execute_form	Perform a domain-specific action that is associated with a form	"Just make the reservation"	Air Travel
		"We got a good photo for H-area"	UAV
report_outcome	Report the outcome of an execute_form operator	"That round trip fare is four hundred three dollars and fifty cents"	Air Travel
		"Currently 7¼ miles out from H-area"	UAV
cancel	Cancel the previous operator	"I would like to complete this leg first"	Air Travel
start_over	Clear the content of a form and restart the sub-task all over again	"Start again"	Map Reading

Table 3.1: A list of task-oriented operators

Operator	Description	Example utterance	Domain
acknowledge	Show understanding of another participant's previous utterance. (can be considered as a backchannel response)	"Right"	Map Reading
		"Roger"	UAV
request_repetition	Request another participant to repeat the previous utterance	"Pardon me"	Bus Schedule
		"Please repeat that again one more time. F-area?"	UAV
greeting	A social utterance at the beginning of the conversation	"Thank you for calling port authority this is Dalisa how may I help you"	Bus Schedule
closing	A social utterance at the end of the conversation	"Okay thank you"	Air Travel

Table 3.2: A list of discourse-oriented operators

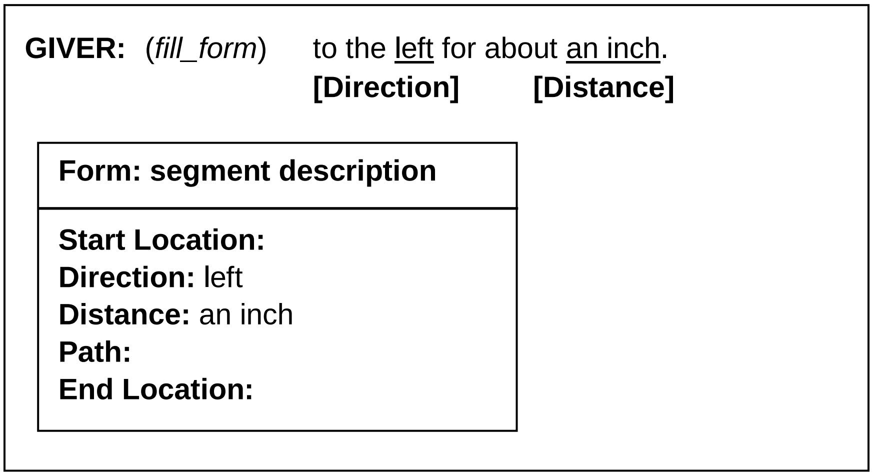


Client: (*fill_form*) i'd like to fly to houston texas
[ArriveCity] [ArriveState]

Form: flight query

DepartCity:
DepartState:
DepartDate:
DepartTime:
ArriveCity: houston
ArriveState: texas

Figure 3.2: A fill_form operator and its effect on the corresponding form in the air travel planning domain



GIVER: (*fill_form*) to the left for about an inch.
[Direction] **[Distance]**

Form: segment description

Start Location:
Direction: left
Distance: an inch
Path:
End Location:

Figure 3.3: A fill_form operator and its effect on the corresponding form in the map reading domain

The effect of a task-oriented operator listed in Table 3.1 can be determined directly from the utterance that is associated with the operator. However, since dialog participants collaborate to achieve a task in a task-oriented dialog, the effects of some utterances on a form may depend on a response from another participant. There are three types of *response-dependent operators*: request, suggest and confirm. Any task-oriented operator can be transformed into a request operator, a suggest operator, or a confirm operator (e.g. request_fill_form, suggest_fill_form, or confirm_fill_form). Examples of response-dependent operators are given in Figure 3.4. The names of the operators are enclosed in parentheses. The dialog in Figure 3.4 is taken from the air travel planning domain and is the same dialog as the one in Figure 3.12.

Client 1:	(*greeting*)	hello
Agent 2:	(*request_fill_form*)	hi people's travel what city would you like to fly to
Client 3:	(*fill_form*)	i'd like to fly to houston texas
		[ArriveCity] [ArriveState]
Agent 4:	(*suggest_fill_form*)	into intercontinental airport or hobby
		[ArriveAirport] [ArriveAirport]
Client 5:	(*fill_form*)	at the intercontinental
		[ArriveAirport]
		…
Agent 12:	(*report_outcome*)	the only flight i have before that that's a non-stop would be on continental airlines that's at
		FlightInfo:[Airline]
		six thirty a.m. arrive houston at eight fifty
		FlightInfo:[DepartTime] FlightInfo:[ArriveCity] FlightInfo:[ArriveTime]
Client 13:	(*fill_form*)	that's okay i'll take that
		FlightInfo:[FlightRef]
Agent 14:	(*confirm_fill_form*)	you'll take the continental flight
		FlightInfo:[Airline]
Client 15:	(*respond*)	yes

Figure 3.4: Examples of response-dependent operators

A dialog participant (a speaker) may request another participant (a listener) to perform a specific operation. For instance, in the second utterance of the dialog in Figure 3.4, an agent requested a client to fill_form with an **ArriveCity**. The client then responded by performing a fill_form operator that filled both **ArriveCity** and **ArriveState** into a flight query form. The request operator by itself doesn't affect the form; nevertheless, it creates an obligation on a listener to perform the requested operation.

When a speaker requests that a listener perform an operation, the speaker doesn't specify all of the parameters of the requested operation; it is up to the listener to choose these parameters. However, the speaker can also suggest operation parameters as illustrated in the forth utterance of the same dialog. The agent suggested two values of an **ArriveAirport** that could be filled into the flight query form. The suggest_fill_form operator by itself didn't fill any of these values into the form as a response from the client was required in order to decide which concept value should be used. The flight query form actually got filled in the fifth utterance by the fill_form operator which is the client's response to the agent's suggest_fill_form operator.

A dialog participant may verify the correctness of an operator and its parameters that have been specified earlier in a dialog with a confirmation utterance. To confirm the correctness of the fill_form operator that was uttered in utterance 13, the agent verified with the client that he/she would like to select the continental flight in the next utterance. With an affirmative response in utterance 15, the information of the selected flight got filled into a flight reservation form. A confirm operator is usually expressed in the form of a yes/no question. An affirmative response carries out the confirmed operation while a negative response discards that operation.

Usually, an utterance or a speaker turn corresponds to one form operator. However, an utterance can correspond to more than one operator if the utterance has more than one distinguishable effect on a form. For instance, a client's turn in Figure 3.5 corresponds to two operators: respond and init_form. In the first part of the turn the client gave a negative response to an agent question about a car reservation while in the second part of the turn the client initiated a discussion about a hotel reservation.

Agent:	(*suggest_init_form*)	do you need a car
Client:	(*respond*)	no
	(*init_form*)	but I do need a hotel

Figure 3.5: An example of a speaker turn that corresponds to more than one operator

I would like to note that even though a form operator describes the role of an utterance in a task-oriented dialog similar to a dialog act, it describes specifically the effect of the utterance on the form representation rather than modeling a speaker underlying intention. This thesis focuses more on the form-based dialog structure representation than on form operators since the form-based representation models domain-dependent components which have to be acquired in every new domain while a list of form operators is domain-independent and is pre-specified.

3.1.3 Task and sub-task decomposition

A list of actions constrains how a task is decomposed into a set of sub-tasks in each domain. An action in the form-based dialog structure representation is defined as a process that uses the information gathered during the conversation to create an outcome that contributes toward the conversation goal. The outcome can be the desired piece of information, such as the inquired departure time from a bus schedule, or a new dialog state that is closer to the desired goal state. However, a process through which a dialog participant acquires each piece of information is not considered as an action. For instance, in a retail domain, obtaining a product name or a quantity from a client is not an action; it fills a form in preparation for eventual execution of the corresponding action. But using the product name and the quantity to make an order is an action since this process makes a dialog reaches its goal state.

Usually, an action is observable from a verbal expression that associates with it (for example, “let me look that up for you”) or from a physical action (for example, an operator types a query and submits it to a database system). Another indication of an action that can be observed is a discussion of the new information obtained from the execution of the action. A retrieved departure time in a bus schedule inquiry domain is an example. However, there are some cases that actions are less noticeable. One example is defining a new term (grounding). A *grounding* action associates a word with its definition or its properties. This knowledge is stored for future reference. If a dialog participant only memorizes the knowledge, neither physical action nor verbal expression occurs. Nevertheless, the process that discusses the term and the information regarding its definition and properties are observable. The grounding process is considered a sub-task which corresponds to a grounding action while the term, its definition, and properties are concepts. The grounding process is further discussed in Section 3.4 and 3.5 when the structures of dialogs in the map reading domain and the UAV flight simulation domain are analyzed respectively.

The definition of an action in the form-based dialog structure representation is different from the one in the plan-based model discussed in Section 2.1.2.3. The action in the plan-based model is a communicative action expressed by an utterance and is usually represented by a speech act. The communicative action is more fine-grained and captures a speaker’s intention rather than a physical action while the action in the form-based representation occurs at the end of an information-exchanging sub-dialog and is defined as a process (usually observable) that uses the exchanged information to create an outcome that contributes toward the conversation goal.

I would like to note that the notions of AccessFrame and attribute proposed by Hardy et al. (2003) and reviewed in Section 2.1.3 are fairly similar to the notions of task and concept in the proposed form-based dialog structure representation. However, the form-based representation offers a richer representation by providing a hierarchical structure of tasks and sub-tasks rather than a flat structure. The form-based representation also provides a more general definition for each dialog structure component and is, therefore, applicable to various types of task-oriented domains. Since Hardy's dialog structure representation is based on the DAMSL annotation scheme, it restricts itself to the information available at the level of utterance (e.g. an AccessFrame only describes a task that corresponds to each utterance) and does not describe an aggregate structure over multiple utterances. The intentional structure in Grosz and Sidner's Theory of discourse structure (Grosz and Sidner, 1986) is also closely related to the structure of tasks and sub-tasks. However, the intentional structure is influenced by discourse segment purposes and their relations rather than domain-specific actions, which capture the characteristics of the tasks more directly, as in case of the form-based dialog structure representation.

To model the structure of a dialog in a new task-oriented domain with the form-based dialog structure representation, a list of tasks, sub-tasks, and concepts in that domain has to be identified. This list can be considered as a domain-dependent tagset and is not pre-specified by the form-based representation but will be identified from in-domain dialogs. The notions of task, sub-task, and concept defined in this section can be regarded as meta-tags and are domain-independent. A summary of task, sub-task, and concept definitions are given in Figure 3.6.

For consistency, I will use the following formatting styles to mark tasks, sub-tasks, concepts, and actions in the rest of this thesis document.

- Task and sub-task types are marked in **bold** with "_" connects all the words together, e.g. **create_an_itinerary** and **grounding**.
- Concept types are marked in **bold** with all the first letter of each word capitalized e.g. **ProductName** and **Quantity**.
- All task, sub-task and concept instances are marked with double quotation marks e.g. "blue" and "green".
- Actions are marked in ***bold italic*** with "_" connects all the words together e.g. ***make_a_flight_reservation***.

Task is a subset of a dialog that has one specific goal

- A dialog corresponds to one task if it has only one goal
- A dialog corresponds to a set of tasks if it has multiple goals and each task may stand alone as a separate dialog
- If as a task requires more than one action, it is decomposed into a set of sub-tasks

Sub-task is a step in a task that contains sufficient information to execute an **action***

- ends after the action that associates with that sub-task is executed or after the outcome of the action has been discussed.

Concept is a word or a group of words that captures

- the information necessary for performing an action
- the information about the outcome of an action

It is also important to distinguish between similar concepts that have different functionalities.

*An **action** is a process that uses related pieces of information stored in a form to

- create a new piece of information (for example, through computation or database retrieval)

or,

- move a conversation to a new state that is closer to the desired goal state

An action **is not** a process through which a dialog participant acquires individual pieces of information.

Figure 3.6: The summary of the form-based dialog structure representation

3.2 Air travel planning domain

A dialog in an air travel planning domain is a conversation between an experienced travel agent and a client arranging a trip that includes plane, hotel and car reservations. A plane ticket reservation is mandatory while hotel and car reservations are optional. A trip can be either a domestic or an international one. In some conversations, a client may have multiple destinations. A corpus of human-human conversations in this domain was collected during the development of the CMU Communicator system. The data collection process is described in (Eskenazi et al., 1999). Figure 3.7 shows a transcript of a recorded conversation in the air travel planning domain. The number that follows the speaker label is an utterance ID. The transcript also includes some noises and fillers (e.g. /UH/ and *PAUSE*) made by both participants and from the environment. Examples of dialog structure analysis discussed in this section are drawn mainly from this dialog.

Client 1: HELLO

Agent 2: /H#/ HI PEOPLE'S TRAVEL WHAT CITY WOULD YOU LIKE TO FLY TO

Client 3: I'D LIKE TO FLY TO HOUSTON TEXAS

Agent 4: INTO INTERCONTINENTAL AIRPORT OR HOBBY

Client 5: AT THE /UH/ INTERCONTINENTAL

Agent 6: AND DEPARTING PITTSBURGH ON WHAT DATE

Client 7: DEPARTING ON FEBRUARY TWENTIETH

Agent 8: WHAT TIME WOULD YOU LIKE TO DEPART PITTSBURGH

Client 9: /UM/ EARLY MORNING NOT BEFORE SEVEN

Agent 10: OKAY U.S. AIRWAYS HAS A NON-STOP AT TEN OH FIVE A.M. YOU'LL ARRIVE INTO HOUSTON AT TWELVE TWENTY P.M.

Client 11: IS THERE AN EARLIER FLIGHT

Agent 12: THE ONLY FLIGHT I HAVE BEFORE THAT THAT'S A NON-STOP WOULD BE ON CONTINENTAL AIRLINES THAT'S AT SIX THIRTY A.M. ARRIVE HOUSTON AT EIGHT FIFTY

Client 13: THAT'S OKAY I'LL TAKE THAT

Agent 14: YOU'LL TAKE THE CONTINENTAL FLIGHT

Client 15: YES

Agent 16: AND WHAT DAY WOULD YOU BE RETURNING

Client 17: ON /UM/ *PAUSE* MONDAY FEBRUARY TWENTY THIRD

Agent 18: WHAT TIME WOULD YOU LIKE TO DEPART HOUSTON

Client 19: AROUND FIVE P.M.

Agent 20: I HAVE A NON-STOP ON CONTINENTAL DEPARTING HOUSTON AT SIX FORTY FIVE P.M. ARRIVING INTO PITTSBURGH AT TEN TWENTY THREE P.M.

Client 21: OKAY

Agent 22: THAT ROUND TRIP FARE IS FOUR HUNDRED THREE DOLLARS AND FIFTY CENTS

Client 23: OKAY

Agent 24: WOULD YOU LIKE ME TO MAKE THE RESERVATION AND DID YOU NEED A CAR

Client 25: YEAH

Agent 26: THEY LEAST EXPENSIVE RATE I HAVE WOULD BE WITH THRIFTY RENTAL CAR AT THE HOUSTON AIRPORT FOR THE WEEKEND RATE OF TWENTY THREE NINETY A DAY

Client 27: OKAY

Agent 28: WOULD YOU LIKE ME TO BOOK THAT CAR FOR YOU

Client 29: YES

Agent 30: OKAY AND WOULD YOU NEED A HOTEL WHILE YOU'RE IN HOUSTON

Client 31: YES

Agent 32: AND WHERE AT IN HOUSTON

Client 33: /UM/ DOWNTOWN

Agent 34: OKAY

Agent 35: DID YOU HAVE A HOTEL PREFERENCE

Client 36: /UM/ ANYTHING HILTON #NOISE# MARRIOTT

Agent 37: I HAVE A MARRIOTT IN DOWNTOWN HOUSTON FOR ONE OH NINE A NIGHT

Client 38: OKAY

Agent 39: WOULD YOU LIKE ME TO BOOK THAT

Client 40: YES

Agent 41: OKAY

Agent 42: WOULD YOU LIKE TO PURCHASE THE TICKET TODAY OR JUST MAKE THE RESERVATION

Client 43: /H#/ JUST MAKE THE RESERVATION

Agent 44: I CAN HOLD THAT TICKET FOR YOU UNTIL TOMORROW AT FIVE P.M. IF YOU COULD PLEASE CALL US BY THEN

Client 45: OKAY

Agent 46: OKAY THANK YOU

Client 47: THANK YOU #CUT IN#

Figure 3.7: An example dialog in the air travel planning domain

A conversation in the air travel domain is an information-accessing task. In this domain, a travel agent helps a client arrange an air-travel itinerary by retrieving flight, hotel, and car rental information from a backend database. In order to do so, the client has to provide the agent with his/her preferences and constraints on the itinerary. Since 1) the conversation goal is achieved by performing domain actions (retrieving information from the database and making a travel reservation), and 2) the client and the agent have to exchange information in order to carry out these actions through dialog, the characteristics of a dialog in the air travel domain match all of the assumptions made by the form-based dialog structure representation. Thus, a dialog in this domain could be modeled by the form-based representation. Detailed analysis of the structure of an air travel planning dialog is given below.

The goal of a conversation in the air travel planning domain is to create an air-travel itinerary for a trip. In each conversation, a client usually has only one trip in mind; therefore, the entire dialog corresponds to a single task, **create_an_itinerary**. However, in some cases, a client may want to arrange several trips in one conversation. In that case, a dialog corresponds to several **create_an_itinerary** tasks; one for each trip.

An air-travel itinerary consists of three types of reservations: a plane ticket reservation, a hotel reservation, and a car rental reservation. A plane ticket reservation is accomplished through a ***make_a_flight_reservation*** action. Similarly, a hotel reservation and a car rental reservation are achieved by a ***make_a_hotel_reservation*** action and a ***make_a_car_reservation*** respectively. Hence, the **create_an_itinerary** task is decomposed into three sub-tasks, one for each type of reservation (flight, hotel, and car). All types of reservations are regarded as sub-tasks instead of tasks even though each of them has a clear goal (to make a specific type of reservation) because they belong to the same itinerary and there is also some dependency between them as will be discussed later in this section. Figure 3.8 shows a decomposition of the task **create_an_itinerary** into three sub-tasks: **reserve_flight**, **reserve_hotel**, and **reserve_car** together with their corresponding forms and actions. The detail of each form is omitted and will be discussed later on.

If a trip has multiple destinations, an itinerary may contain more than one plane ticket reservation as well as multiple hotel and car reservations. In that case, a dialog may contain multiple instances of **reserve_flight**, **reserve_hotel**, and **reserve_car** sub-tasks. On the other hand, in some conversations, **reserve_hotel**, and/or **reserve_car** sub-tasks are optional. Only a **reserve_flight** sub-task is mandatory in this domain.

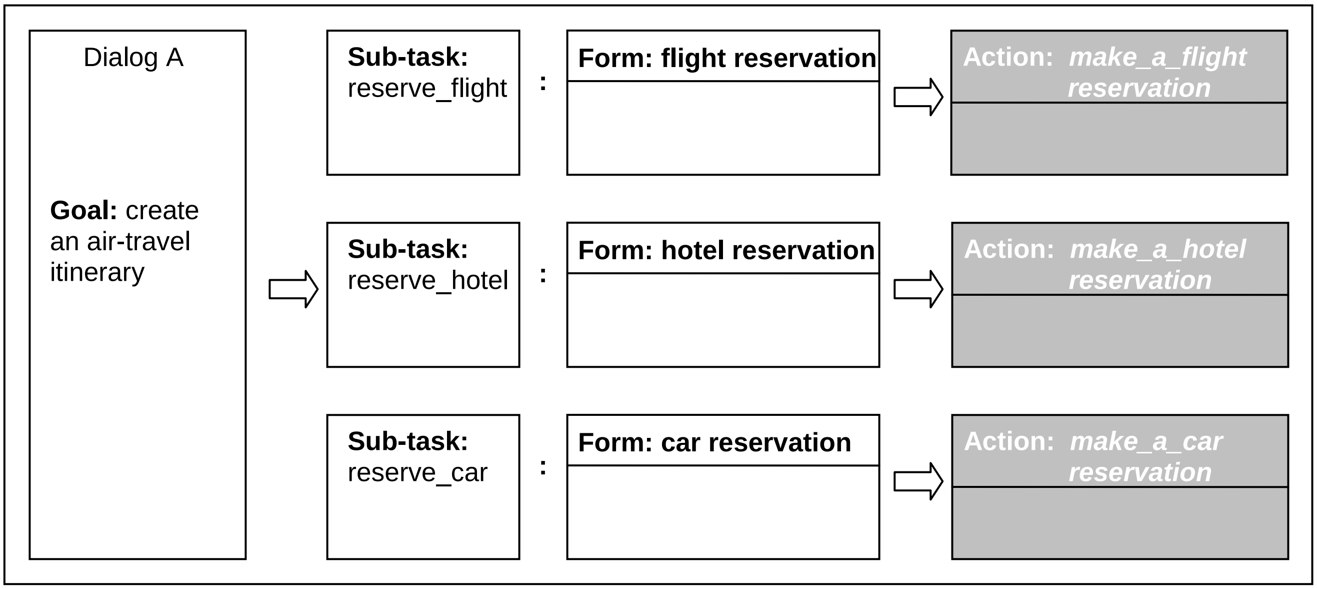


Figure 3.8: A task, sub-tasks and their corresponding forms and actions in the air travel planning domain

A **reserve_flight** sub-task can be further decomposed. To make a flight reservation for a round trip, an agent needs to know the flight that a client would like to take for each leg of the trip (i.e. a departure flight and a return flight). The client must provide criteria of the preferred flight for each leg to the agent who then retrieves flight(s) that matches the given criteria from a database. Therefore, a **reserve_flight** sub-task is then decomposed into two **query_flight_info** sub-subtasks, one for each ***retrieve_flight_fromDB*** action required for each leg of the trip. The criteria for retrieving a desired flight include a **DepartureCity**, an **ArrivalCity**, a **DepartureDate**, an **ArrivalDate**, **a DepartureTime**, an **ArrivalTime**, an **Airline**, etc. These are concepts in a flight query form associated with a **query_flight_info** sub-subtask. Since not all of the criteria have to be specified in order to retrieve information from a database, some concepts in the flight query form may not get filled.

In the dialog in Figure 3.7, a client would like to reserve a round trip flight from Pittsburgh to Houston. Two flight query forms, one for each leg of the round trip reservation, are shown in Figure 3.9. Only the slots that are discussed in the dialog are presented in the forms. In some examples, detailed representation and annotation are omitted for a display purpose. Some concept names may be shortened and structured concept components may be excluded. For example, a **DepartureDate** is a structure concept consists of a **Month** and a **Date**, but for simplicity we only represent it as "**DepartDate:** February twentieth" instead of "**DepartDate: Month:** February **Date:**

twentieth" in the flight query form. A dialog structure annotation of the conversation in Figure 3.7 is given at the end of this section.

The result of a ***retrieve_flight_fromDB*** action is the information of the flight(s) that matches the given criteria. "CONTINENTAL AIRLINES THAT'S AT SIX THIRTY A.M. ARRIVE HOUSTON AT EIGHT FIFTY" excerpted from Utterance 12 in Figure 3.7 is one example of the result. This piece of information can be regarded as a structured concept **FlightInfo**, which composes of an **Airline** "Continental", a **DepartTime** "six thirty a.m.", an **ArriveCity** "Houston", and an **ArriveTime** "eight fifty". A **FlightInfo** is a structured concept that refers to a particular flight and will be used by a travel agent to make a flight reservation.

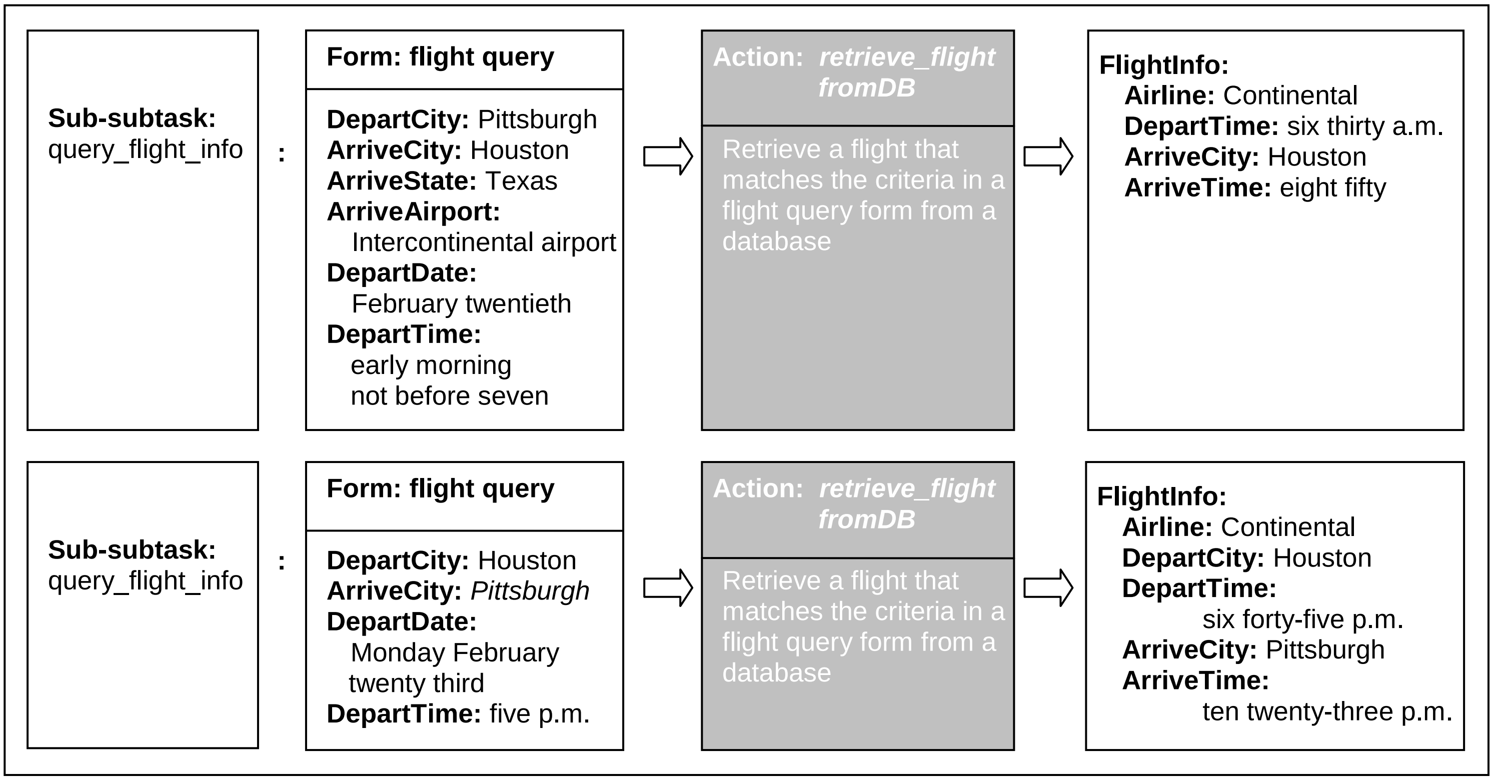
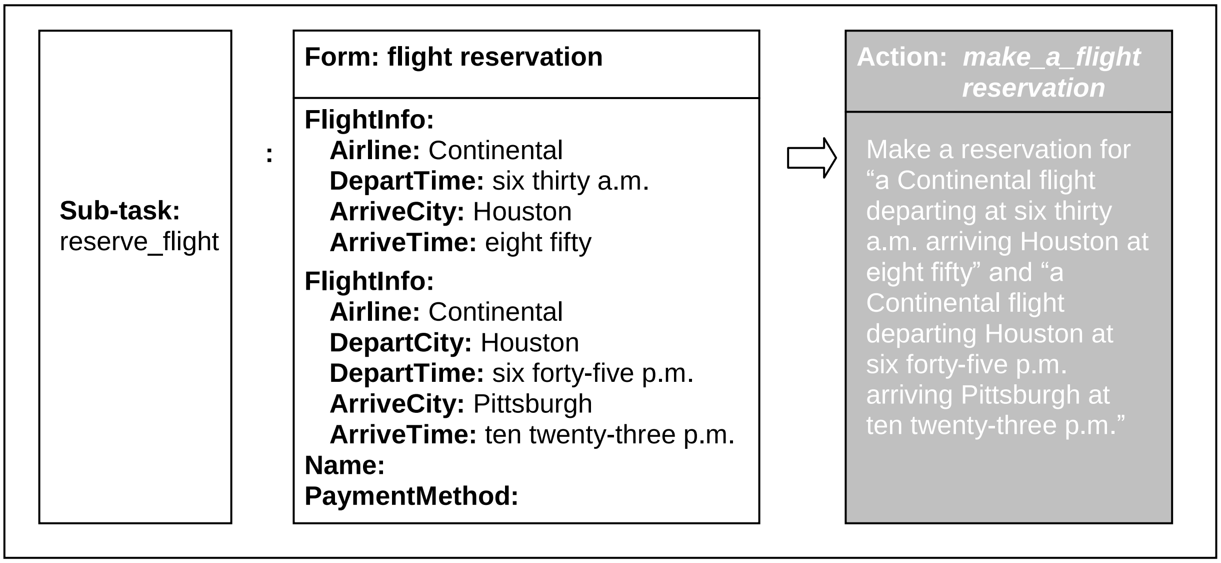

Sub-subtask: query_flight_info
:
Form: flight query
DepartCity: Pittsburgh
ArriveCity: Houston
ArriveState: Texas
ArriveAirport: Intercontinental airport
DepartDate: February twentieth
DepartTime: early morning not before seven
Action: retrieve_flight fromDB
Retrieve a flight that matches the criteria in a flight query form from a database
FlightInfo:
Airline: Continental
DepartTime: six thirty a.m.
ArriveCity: Houston
ArriveTime: eight fifty
Sub-subtask: query_flight_info
:
Form: flight query
DepartCity: Houston
ArriveCity: Pittsburgh
DepartDate: Monday February twenty third
DepartTime: five p.m.
Action: retrieve_flight fromDB
Retrieve a flight that matches the criteria in a flight query form from a database
FlightInfo:
Airline: Continental
DepartCity: Houston
DepartTime: six forty-five p.m.
ArriveCity: Pittsburgh
ArriveTime: ten twenty-three p.m.

Figure 3.9: Flight query forms, their corresponding actions and the outcomes for a round trip reservation

In many cases, there is more than one flight that matches a client's criterion. The client needs to select only one flight from a list of flights returned by a ***retrieve_flight_fromDB*** action. After the client select the desired flight, the **FlightInfo** of the selected flight gets filled into a flight reservation form. When all required **FlightInfos** for that trip are obtained, e.g. two for a round trip, an agent will perform a ***make_a_flight_reservation*** action. Figure 3.10 shows a flight reservation form that contains two **FlightInfos**, one for each leg of the trip, retrieved by two ***retrieve_flight_fromDB*** actions in Figure 3.9. In some conversations, a flight reservation form may contain additional concepts such as a client's **Name** and a **PaymentMethod**. These additional concepts may not be discussed in some dialogs such as the one in Figure 3.7 because an agent may already have that information in a client's profile.



Sub-task: reserve_flight

:

Form: flight reservation

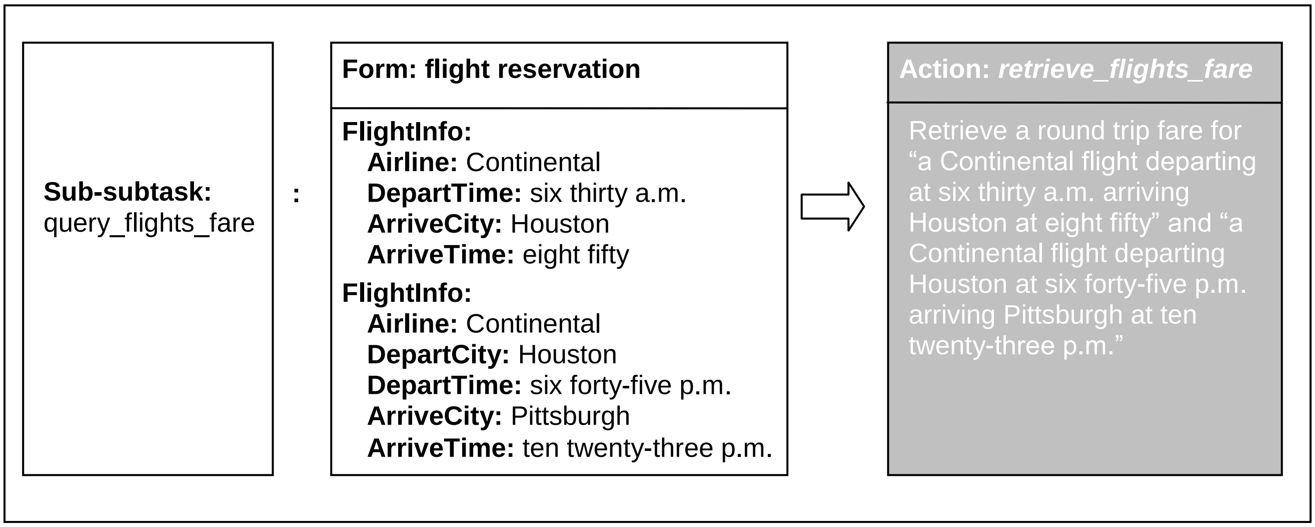
FlightInfo:
Airline: Continental
DepartTime: six thirty a.m.
ArriveCity: Houston
ArriveTime: eight fifty
FlightInfo:
Airline: Continental
DepartCity: Houston
DepartTime: six forty-five p.m.
ArriveCity: Pittsburgh
ArriveTime: ten twenty-three p.m.
Name:
PaymentMethod:

Action: ***make_a_flight reservation***

Make a reservation for "a Continental flight departing at six thirty a.m. arriving Houston at eight fifty" and "a Continental flight departing Houston at six forty-five p.m. arriving Pittsburgh at ten twenty-three p.m."

Figure 3.10: A reserve_flight sub-task and the corresponding form and action for a round trip reservation

Another action that may occur in a **reserve_flight** sub-task is a ***retrieve_flights_fare*** action. A client may want to know the ticket price before making a reservation. Normally a ticket fare is based on all of the flights in the itinerary together. For example, a round trip fare is usually cheaper than the summation of two one-way fares. Therefore, the ***retrieve_flights_fare*** action requires information of all of the flights in an itinerary. Since a **FlightInfo** is an outcome of a ***retrieve_flight_fromDB*** action, a **query_flights_fare** sub-subtask, which corresponds to a ***retrieve_flights_fare*** action, occurs after all **query_flight_info** sub-subtasks. Figure 3.11 shows a fare query form for retrieving a ticket fare of a round trip ticket in Figure 3.9.



Sub-subtask: query_flights_fare

:

Form: flight reservation

FlightInfo:
Airline: Continental
DepartTime: six thirty a.m.
ArriveCity: Houston
ArriveTime: eight fifty

FlightInfo:
Airline: Continental
DepartCity: Houston
DepartTime: six forty-five p.m.
ArriveCity: Pittsburgh
ArriveTime: ten twenty-three p.m.

Action: *retrieve_flights_fare*

Retrieve a round trip fare for "a Continental flight departing at six thirty a.m. arriving Houston at eight fifty" and "a Continental flight departing Houston at six forty-five p.m. arriving Pittsburgh at ten twenty-three p.m."

Figure 3.11: A fare query form and its corresponding action.

Since actions and the corresponding forms for **reserve_hotel**, and **reserve_car** sub-tasks are quite similar, I only discuss a **reserve_hotel** sub-task in this section. In a **reserve_hotel** sub-task, a client first specifies the criteria of the hotel room that he/she would like to reserve such as a **HotelName** and an **Area** of a city. These concepts are filled into a hotel query form which is then used by an agent to retrieve hotel room(s) that matches the given criteria from a database via a ***retrieve_hotel_info_fromDB*** action. An example of a hotel query form is given in Figure 3.13 (f). **query_hotel_info** is a sub-subtask under a **reserve_hotel** sub-task and is associated with a ***retrieve_hotel_info_fromDB*** action. Similar to a **FlightInfo**, a **HotelInfo** is a structured concept that contains information about a particular hotel room retrieved from a database. The client has to select only one hotel room from a list of hotel rooms retrieved by a ***retrieve_hotel_info_fromDB*** action. The selected **HotelInfo** gets filled into a hotel reservation form as illustrated in Figure 3.13 (g).

A **reserve_hotel** sub-task is less complex than a **reserve_flight** sub-task. A hotel for each stop in an itinerary can be reserved separately while all of the flights should be reserved together as in the case of a round trip. Hence, only one **HotelInfo** is required in each hotel reservation form while a **FlightInfo** of every leg in the itinerary is required for a flight reservation form. Hotel fare is also associated with an individual hotel room; therefore, it can be retrieved from a database by the same ***retrieve_hotel_info_fromDB*** action as other information in a **HotelInfo**. A separate ***retrieve_hotel_fare*** action is not necessary.

Table 3.3 summarizes the structure of a dialog in the air travel planning domain. The hierarchical structure of tasks and sub-tasks is indicated by the numbering in the first column. Examples of related concepts for each task or sub-task are presented in the last column.

ID	Name	Associated action	Related concepts
Task 1	create_an_itinerary		
Sub-task 1.1	reserve_flight	make_a_flight_reservation	FlightInfos, Name, PaymentMethod
Sub-subtask 1.1.1	query_flight_info	retrieve_flight_fromDB	DepartCity, ArriveCity, DepartTime, DepartDate
Sub- subtask 1.1.2	query_flights_fare	retrieve_flights_fare	FlightInfos
Sub-task 1.2	reserve_hotel	make_a_hotel_reservation	HotelInfo, Name, PaymentMethod
Sub- subtask 1.2.1	query_hotel_info	retrieve_hotel_info_fromDB	HotelName, Area
Sub-task 1.3	reserve_car	make_a_car_reservation	CarInfo, Name, PaymentMethod
Sub- subtask 1.3.1	query_car_info	retrieve_car_info_fromDB	RentalCompany, CarSize

Table 3.3: Task, sub-tasks and their corresponding actions and concepts in the air travel planning domain

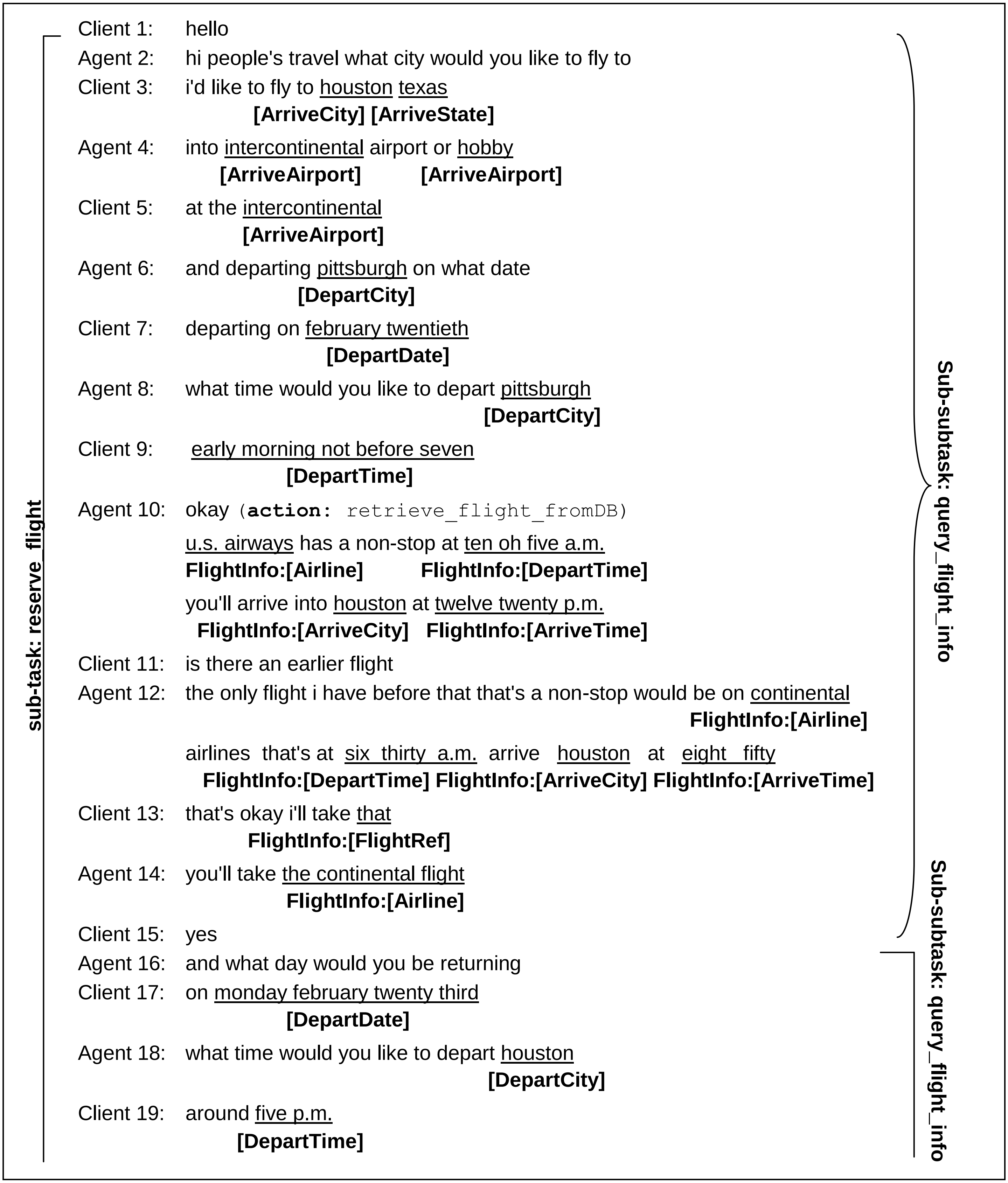
Figure 3.12 illustrates a dialog structure annotation for the conversation in Figure 3.7. In this example, the entire dialog corresponds to one task **create_an_itinerary**; therefore, only sub-task and sub-subtask boundaries are illustrated.

The following notions are used to illustrate the structure of a dialog.

- The bracket on the left shows the boundaries of a sub-task while the bracket on the right shows the boundaries of a sub-subtasks
- An instance of a concept is underlined and the concept name is enclosed in a square bracket underneath it.
- The name of the structured concept is placed on the left-handed side next to its component names. For simplicity the annotation of some structured concept components are excluded. For example, the full annotation for “february twentieth” is **DepartDate:[[Month][Date]]**.
- (action: …) indicates approximately when an action occurs in a conversation

This dialog consists of three sub-tasks: **reserve_flight** (round trip), **reserve_car**, and **reserve_hotel**. The criteria for an out-bound flight and an in-bound flight are captured in the flight query forms in Figure 3.13 (a) and (b) respectively. The flights that a client selected are presented in a flight reservation form in Figure 3.13 (c) under **FlightInfo** concepts. Both concepts were used to retrieve a **Fare** in a **query_flights_fare** sub-subtask. A separate fare query form is not presented. The make reservation part of the flight reservation was interrupted by the discussion of the car reservation. It was resumed at the end of the conversation. Information about a client's name and a payment method were omitted in this dialog.

There are a lot of dependencies among a flight reservation, a car reservation, and a hotel reservation. An agent did not need more information about a rental car from the client as all of the concepts required in a car query form. For instance, **PickUpDate** and **PickUpTime** could be inferred from the information of the selected flights. These implicit concepts are marked in italic in the car query form in Figure 3.13 (d). The information of the car that the client reserved is shown in Figure 3.13 (e). For the hotel reservation, the agent inquired a preferred **Area** and **HotelName** from the client in addition to a **CheckInDate** and a **CheckOutDate** that can be inferred from the selected flights. The hotel query form and the hotel reservation form are shown in Figure 3.13 (f) and (g) respectively. Implication of implicit concepts and interrupted sub-tasks on dialog structure learning is discussed in Section 3.8.


sub-task: reserve_flight
Sub-subtask: query_flight_info
Sub-subtask: query_flight_info

Client 1: hello

Agent 2: hi people's travel what city would you like to fly to

Client 3: i'd like to fly to houston texas
[ArriveCity] [ArriveState]

Agent 4: into intercontinental airport or hobby
[ArriveAirport] **[ArriveAirport]**

Client 5: at the intercontinental
[ArriveAirport]

Agent 6: and departing pittsburgh on what date
[DepartCity]

Client 7: departing on february twentieth
[DepartDate]

Agent 8: what time would you like to depart pittsburgh
[DepartCity]

Client 9: early morning not before seven
[DepartTime]

Agent 10: okay (**action:** retrieve_flight_fromDB)

u.s. airways has a non-stop at ten oh five a.m.
FlightInfo:[Airline] **FlightInfo:[DepartTime]**

you'll arrive into houston at twelve twenty p.m.
FlightInfo:[ArriveCity] **FlightInfo:[ArriveTime]**

Client 11: is there an earlier flight

Agent 12: the only flight i have before that that's a non-stop would be on continental
FlightInfo:[Airline]

airlines that's at six thirty a.m. arrive houston at eight fifty
FlightInfo:[DepartTime] **FlightInfo:[ArriveCity]** **FlightInfo:[ArriveTime]**

Client 13: that's okay i'll take that
FlightInfo:[FlightRef]

Agent 14: you'll take the continental flight
FlightInfo:[Airline]

Client 15: yes

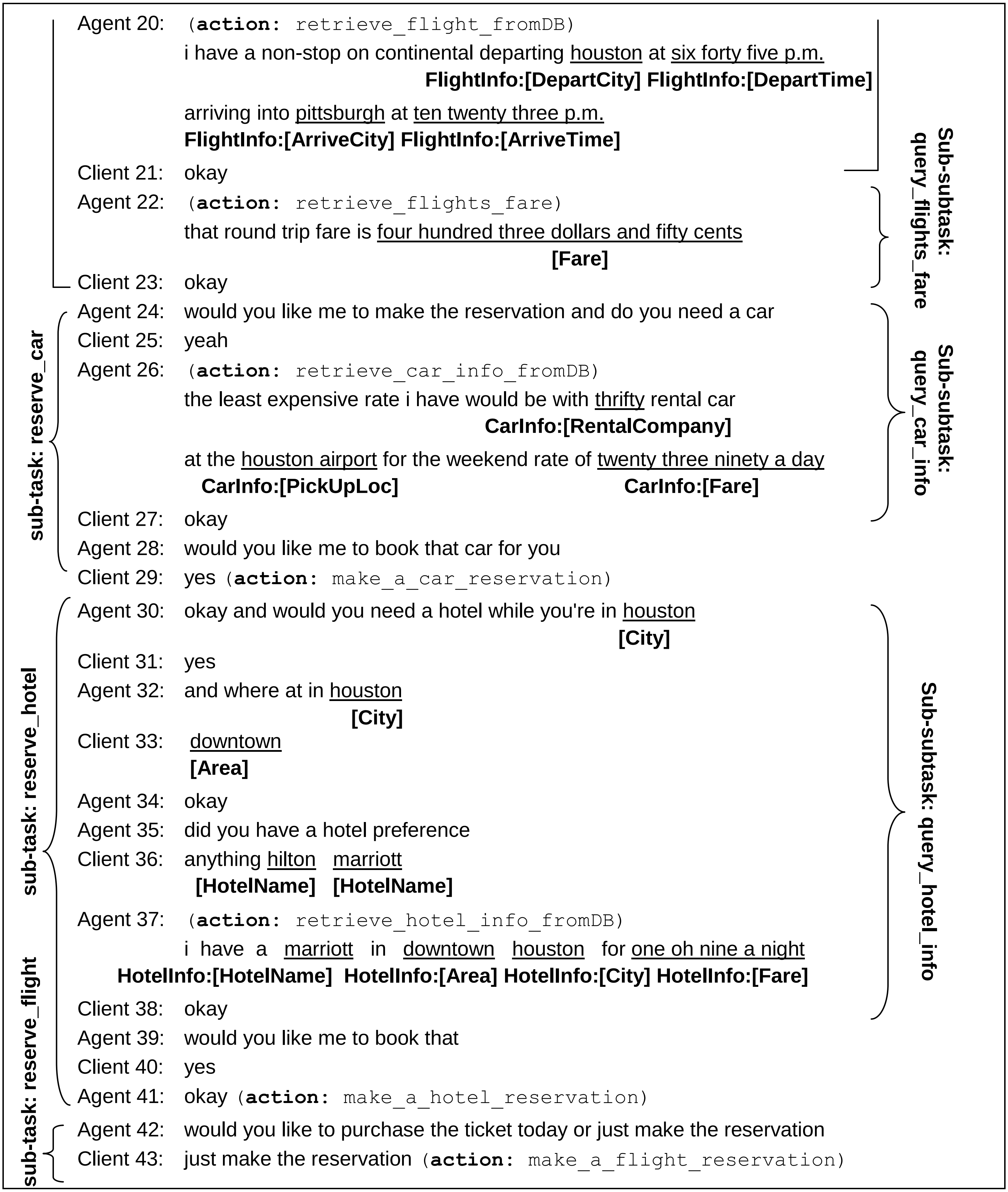
Agent 16: and what day would you be returning

Client 17: on monday february twenty third
[DepartDate]

Agent 18: what time would you like to depart houston
[DepartCity]

Client 19: around five p.m.
[DepartTime]

Figure 3.12: An example of dialog structure annotation in the air travel domain


Sub-subtask: query_flights_fare
sub-task: reserve_car
Sub-subtask: query_car_info
sub-task: reserve_hotel
Sub-subtask: query_hotel_info
sub-task: reserve_flight

Agent 20: (**action:** `retrieve_flight_fromDB`)
i have a non-stop on continental departing houston at six forty five p.m.
FlightInfo:[DepartCity] FlightInfo:[DepartTime]
arriving into pittsburgh at ten twenty three p.m.
FlightInfo:[ArriveCity] FlightInfo:[ArriveTime]
Client 21: okay
Agent 22: (**action:** `retrieve_flights_fare`)
that round trip fare is four hundred three dollars and fifty cents
[Fare]
Client 23: okay
Agent 24: would you like me to make the reservation and do you need a car
Client 25: yeah
Agent 26: (**action:** `retrieve_car_info_fromDB`)
the least expensive rate i have would be with thrifty rental car
CarInfo:[RentalCompany]
at the houston airport for the weekend rate of twenty three ninety a day
CarInfo:[PickUpLoc] **CarInfo:[Fare]**
Client 27: okay
Agent 28: would you like me to book that car for you
Client 29: yes (**action:** `make_a_car_reservation`)
Agent 30: okay and would you need a hotel while you're in houston
[City]
Client 31: yes
Agent 32: and where at in houston
[City]
Client 33: downtown
[Area]
Agent 34: okay
Agent 35: did you have a hotel preference
Client 36: anything hilton marriott
[HotelName] [HotelName]
Agent 37: (**action:** `retrieve_hotel_info_fromDB`)
i have a marriott in downtown houston for one oh nine a night
HotelInfo:[HotelName] HotelInfo:[Area] HotelInfo:[City] HotelInfo:[Fare]
Client 38: okay
Agent 39: would you like me to book that
Client 40: yes
Agent 41: okay (**action:** `make_a_hotel_reservation`)
Agent 42: would you like to purchase the ticket today or just make the reservation
Client 43: just make the reservation (**action:** `make_a_flight_reservation`)

Figure 3.12: An example of dialog structure annotation in the air travel domain (cont.)

Form: flight query

DepartCity: Pittsburgh
ArriveCity: Houston
ArriveState: Texas
ArriveAirport: Intercontinental airport
DepartDate: February twentieth
DepartTime: early morning not before seven

(a)

Form: flight query

DepartCity: Houston
ArriveCity: *Pittsburgh*
DepartDate: Monday February twenty third
DepartTime: five p.m.

(b)

Form: flight reservation

FlightInfo:
Airline: Continental
DepartTime: six thirty a.m.
ArriveCity: Houston
ArriveTime: eight fifty
FlightInfo:
Airline: Continental
DepartCity: Houston
DepartTime: six forty-five p.m.
ArriveCity: Pittsburgh
ArriveTime: ten twenty-three p.m.
Fare: four hundred three dollars and fifty cents
Name:
PaymentMethod:

(c)

Form: car query

PickUpLoc: *Houston*
PickUpDate: *February twentieth*
PickUpTime: *after eight fifty*
DropOffLoc: *Houston*
DropOffDate: *February twenty third*
DropOffTime: *before six forty-five p.m.*

(d)

Form: car reservation

CarInfo:
PickUpLoc: Houston airport
RentalCompany: Thrifty
Fare: twenty three ninety a day
Name:
PaymentMethod:

(e)

Form: hotel query

City: Houston
Area: downtown
HotelName: Hilton, Marriott
CheckInDate: *February twentieth*
CheckOutDate: *February twenty third*

(f)

Form: hotel reservation

HotelInfo:
City: Houston
Area: downtown
HotelName: Marriott
Fare: one oh nine a night
Name:
PaymentMethod:

(g)

Figure 3.13: All of the forms that correspond to the dialog structure annotation in the air travel domain presented in Figure 3.12

3.3 Bus schedule inquiry domain

Conversations in the bus schedule inquiry domain are taken from the Let's Go corpus (Raux et al., 2003). This corpus is a collection of calls to the Pittsburgh Port Authority Transit system enquiring about the bus schedule and other related issues such as lost-and-found and driver behavior complaint. Each conversation is a telephone conversation between a help desk operator and a client. We selected those conversations that involved enquiries about the bus schedule for the analysis. An example of dialogs in this domain is shown in Figure 3.14. The number that follows the speaker label is an utterance ID. The transcript also includes some noises and fillers (e.g. /um/ and /feed/) made by both participants and from the environment.

Operator	1:	thank you for calling port authority this is dalisa how may i help you
Client	2:	yeah /feed/ i'm looking for a 41E leaving downtown pittsburgh /um/ around three o'clock
Operator	3:	there would be one due at two_forty_five three_seventeen or three_forty_five
Client	4:	ok thank you

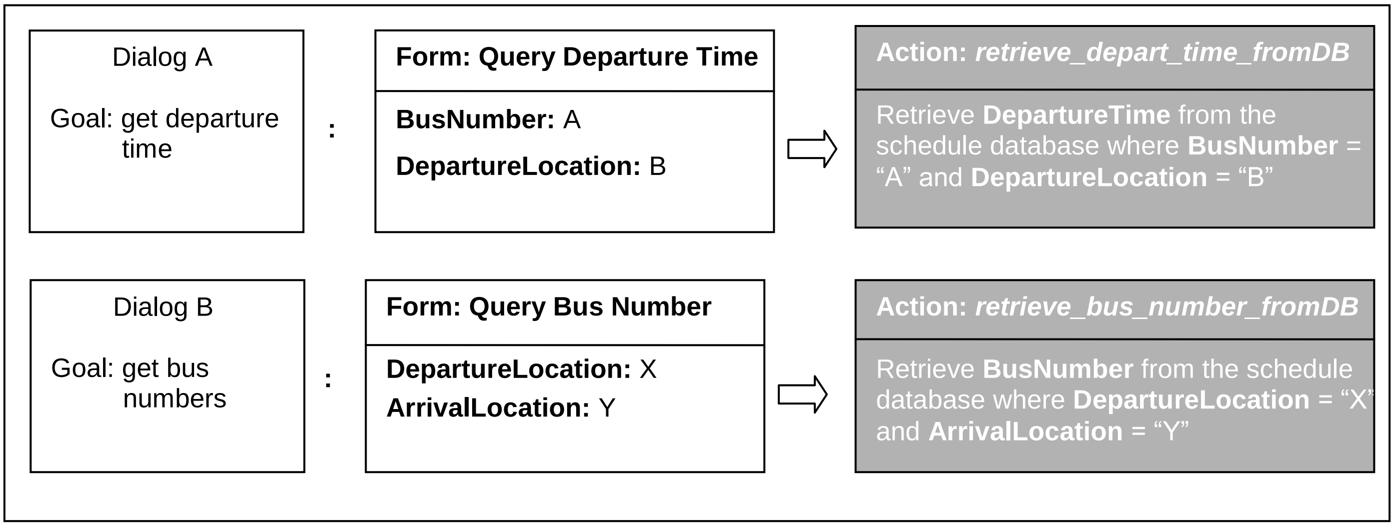
Figure 3.14: An example dialog in the bus schedule enquiry domain

A conversation in the bus schedule inquiry domain is considered an information-accessing task similar to a conversation in the air travel planning domain discussed in the previous section. In this domain, an operator helps a client find the desired bus information by looking up the information from the bus schedule. In order to do so, the client has to provide enough criteria to do the search. Since 1) the conversation goal is achieved through the execution of a domain action (looking up the information from the bus schedule), and 2) the client and the operator have to exchange information required to perform this action through dialog, the characteristics of a dialog in the bus schedule inquiry domain match all of the assumptions made by the form-based dialog structure representation. Therefore, a dialog in this domain could be modeled by the form-based representation. Detailed analysis of the structure of a bus schedule inquiry dialog is given below.

The goal of a conversation in this domain is to obtain information about the bus schedule. However, unlike a conversation in the air travel planning domain, a client's specific goal may vary from dialog to dialog depending on the specific piece of information that the client would like to obtain from the bus schedule (e.g., the time that a specific bus leaves a given bus stop location, or the numbers of the buses that run between two locations).

To answer a client's question, an operator looks up the inquired pieces of information from the bus schedule. In order to do so, the operator needs to gather enough information about the question from the client. For example, to answer a client's question about a departure time, an operator needs a **BusNumber** and a **DepartureLocation** from the client. These pieces of information are slots in a query form as shown in Figure 3.15. After getting all of the necessary information, the operator retrieves the departure time from a database and then informs the retrieved information to the client. The outcome of the action ***retrieve_dept_time_fromDB***, which is a **DepartureTime**, is also a concept. Only one simple action, "retrieve information from the database", is required to accomplish a conversation goal; therefore, in this domain there is no need to decompose a task into sub-tasks.

If a dialog has a different goal, the corresponding action and form will be different. Dialog B shows another goal, "get bus numbers"; therefore, the corresponding form contains different slot items.



Retrieve **DepartureTime** from the schedule database where **BusNumber** = "A" and **DepartureLocation** = "B"

Retrieve **BusNumber** from the schedule database where **DepartureLocation** = "X" and **ArrivalLocation** = "Y"

Figure 3.15: Actions and the associated forms in the bus schedule inquiry domain

There are several types of tasks in this domain depending on the type of information that a client would like to obtain from the bus schedule. Two types of tasks are shown in Table 3.4. The concepts shown in the query forms in Figure 3.15 and in Table 3.4 are a minimal set of concepts required for each action. Human-human conversations are more flexible than a query language; therefore, for the same type of task a slightly different set of concepts may be used. A client may specify additional information to constraint the search as shown in the next example. On the other hand, the concept that is commonly known by both participants can be omitted; for instance, the departure date can be assumed to be the current date.

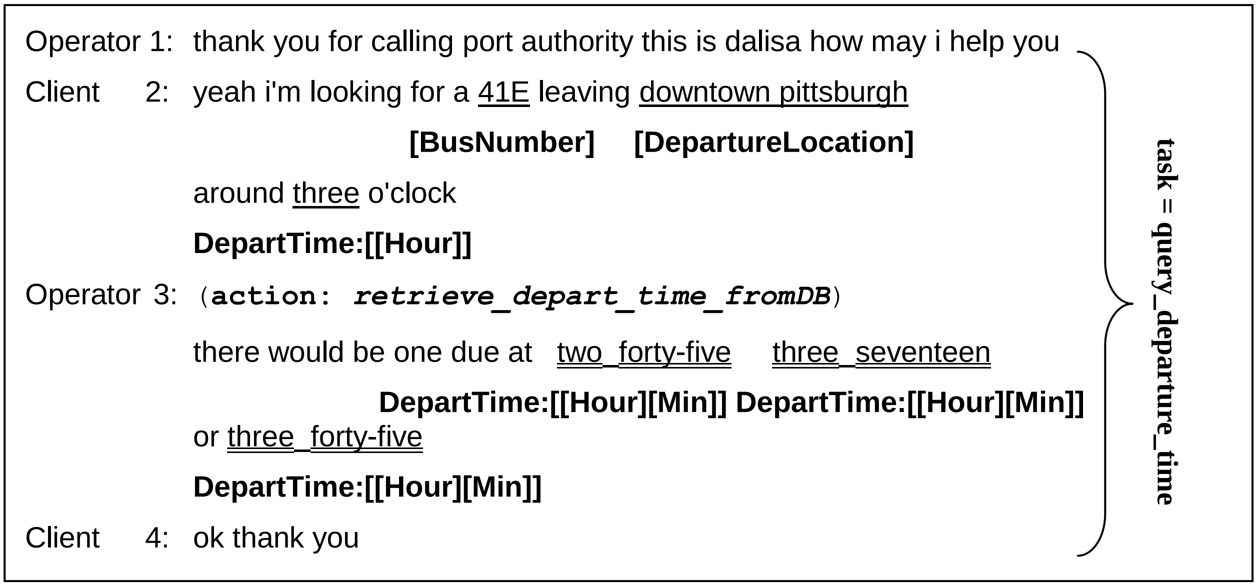
ID	Name	Associated action	Related concepts
Task 1	query_departure_time	retrieve_dept_time_fromDB	BusNumber, DepartureLocation
Task 2	query_bus_number	retrieve_bus_number_fromDB	DepartureLocation, ArrivalLocation

Table 3.4: Examples of tasks and the corresponding actions and concepts in the bus schedule inquiry domain

Figure 3.16 illustrates a dialog structure annotation for the conversation shown in Figure 3.14. The goal of this conversation it to get the departure time for the bus of interest. The client also specified an approximated **DepartureTime** that he was interested in to reduce the scope of the search. This concept is an optional concept and is not listed in Table 3.4. The corresponding form of this conversation is shown in Figure 3.17.

The following notions are used to illustrate the structure of a dialog. Theses notions are similar to the ones used in Figure 3.12.

- The curly bracket on the right shows the boundaries of a task.
- An instance of a concept is underlined and the concept name is enclosed in a square bracket underneath it.
- The name of the structured concept is placed on the left-handed side next to its component names. A **DeptTime** (a shorten form of a **DepartureTime**) is a structured concept consists of an **Hour** and a **Min** (minute).
- (action: …) indicates approximately when an action occurs in a conversation .



Operator 1: thank you for calling port authority this is dalisa how may i help you

Client 2: yeah i'm looking for a 41E leaving downtown pittsburgh

[BusNumber] **[DepartureLocation]**

around three o'clock

DepartTime:[[Hour]]

Operator 3: (**action:** ***retrieve_depart_time_fromDB***)

there would be one due at two_forty-five three_seventeen

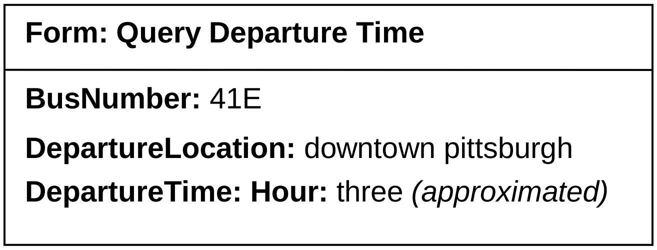
DepartTime:[[Hour][Min]] DepartTime:[[Hour][Min]]

or three_forty-five

DepartTime:[[Hour][Min]]

Client 4: ok thank you

Figure 3.16: An example of dialog structure annotation in the bus schedule inquiry domain



Form: Query Departure Time

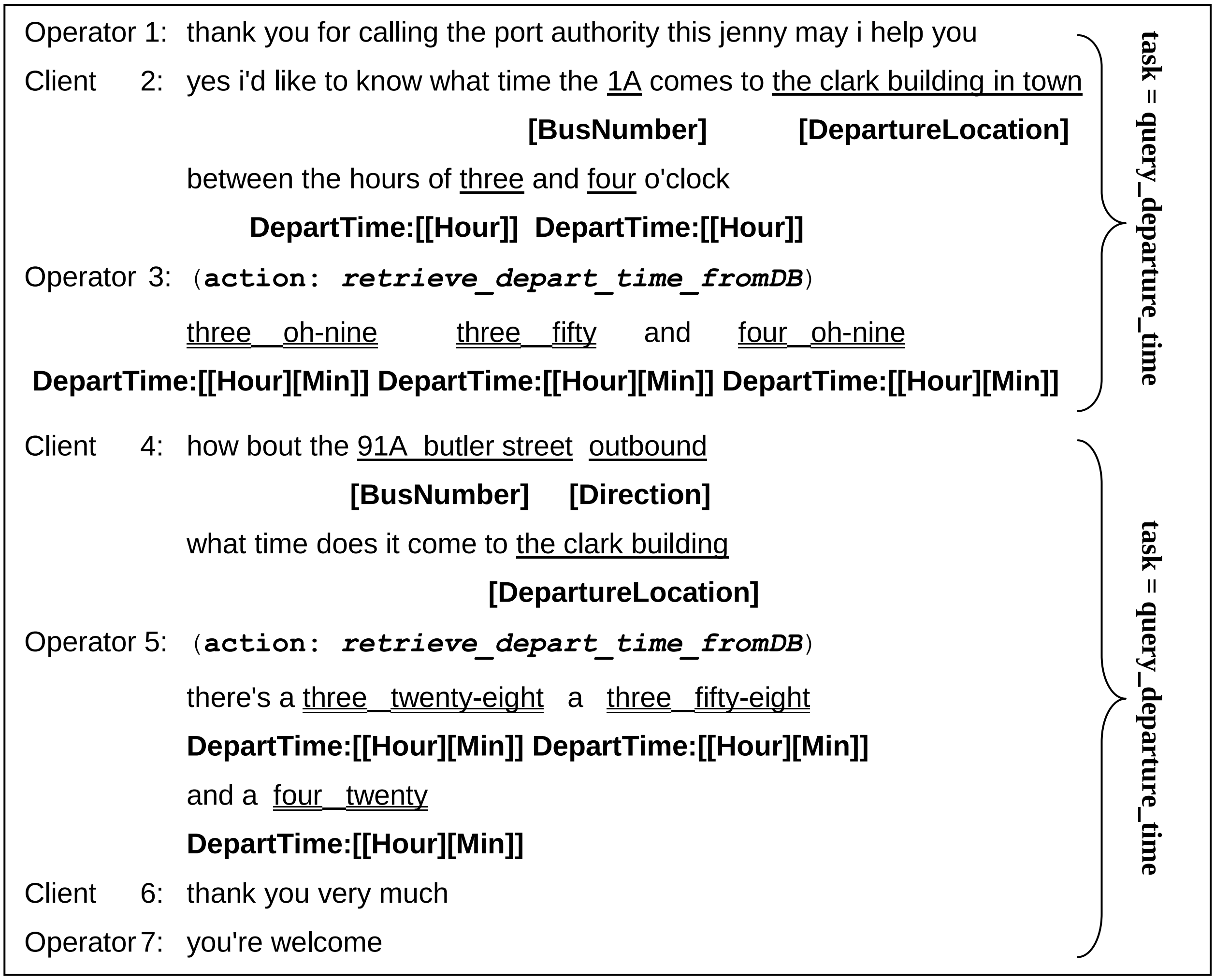
BusNumber: 41E

DepartureLocation: downtown pittsburgh

DepartureTime: Hour: three *(approximated)*

Figure 3.17: A form query_departure_time in the bus schedule inquiry domain

Most of the dialogs in this domain have only one goal; therefore, the entire dialog corresponds to one task as illustrated in Figure 3.16. However, if a client would like to ask several questions about the bus schedule, that dialog has multiple goals, one for each question, and hence corresponds to multiple tasks. Unlike all sub-tasks in the air travel planning domain, each bus schedule query in this domain is independent from each other as all required information is independent; therefore, it can be a separate task. In the following conversation, a client would like to know a **DepartureTime** of two different buses. So, this dialog corresponds to two **query_departure_time** tasks as illustrated by the annotation in Figure 3.18.



Operator 1: thank you for calling the port authority this jenny may i help you

Client 2: yes i'd like to know what time the 1A comes to the clark building in town

[BusNumber] **[DepartureLocation]**

between the hours of three and four o'clock

DepartTime:[[Hour]] **DepartTime:[[Hour]]**

Operator 3: (**action:** ***retrieve_depart_time_fromDB***)

three oh-nine three fifty and four oh-nine

DepartTime:[[Hour][Min]] **DepartTime:[[Hour][Min]]** **DepartTime:[[Hour][Min]]**

Client 4: how bout the 91A butler street outbound

[BusNumber] **[Direction]**

what time does it come to the clark building

[DepartureLocation]

Operator 5: (**action:** ***retrieve_depart_time_fromDB***)

there's a three twenty-eight a three fifty-eight

DepartTime:[[Hour][Min]] **DepartTime:[[Hour][Min]]**

and a four twenty

DepartTime:[[Hour][Min]]

Client 6: thank you very much

Operator 7: you're welcome

Figure 3.18: An example of dialog structure annotation for a dialog that contains two tasks

3.4 Map reading domain

Dialogs in the map reading domain are taken from the HCRC Map Task corpus (Anderson et al., 1991), collected at University of Edinburgh and University of Glasgow. Each dialog is a conversation between two participants: a route giver and a route follower. Both of them have maps of artificial places; however, only the route giver's map has a route printed on it while the route follower's one does not. The goal of the conversation is to have the follower reproduce the route on the giver's map solely through a dialog. The maps may have differences that complicate the task. The task in this domain can be considered as a problem-solving task. In order for the follower to reproduce the route on his/her map, the giver has to describe how to draw it to the follower. Since 1) the conversation goal is achieved by performing a domain action, drawing a route, and 2) the route giver and the route follower have to communicate the description of the route in order to draw it through dialog, the characteristics of a dialog in the map reading domain match all of the assumptions made by the form-based dialog structure representation. Thus, a dialog in this domain could be modeled by the form-based representation. The remainder of this section provides detailed analysis of the structure of a map reading dialog.

A transcript of a conversation in the map reading domain is shown in Figure 3.19 and the corresponding maps are shown in Figure 3.20 with a route giver's map on the left and a route follower's map on the right. Since conversations in this domain are usually long, around 150 utterances in average, only the first part of the conversation is presented. The utterances labeled with "GIVER" belong to the route giver while the utterances labeled with "FOLLOWER" belong to the route follower. The number that follows the speaker label is an utterance ID. The transcript also includes additional markers such as "--" for disfluency and "…" for discontinuity in the recorded speech. A dialog structure annotation of this conversation is given at the end of this section. For the purpose of illustration, some examples of dialog structure analysis discussed in this section are taken from other sources.

The goal of a conversation in the map reading domain is to reproduce the giver's route on the follower's map. Since drawing an entire route is a complicated action, a **draw_a_route** task can be divided into a series of similar sub-tasks; each of them focuses on a small segment of the route. Each **draw_a_segment** sub-task corresponds to one ***draw_a_segment*** action which draws a small segment of the route on the follower's map.

To draw a segment of the route on his/her map, the follower needs a detailed description of the segment from the route giver. Information in the description may include a **StartLocation**, a **Direction**, a **Distance**, and an **EndLocation**. These items of information are concepts in a **draw_a_segment** sub-task. To describe a rather complex route segment, the route giver may also include the landmarks found along the way in a segment description. Information about locations in a segment description that are not a **StartLocation** and an **EndLocation** is capture by a concept **Path**. All location types (**StartLocation**, **EndLocation** and **Path**) can be described in terms of an absolute position on the map such as "at the left corner of the page" or in relative to a known landmark on the map such as "above the waterfall". A relative location can be considered a structure concept which consists of a **Relation** and a **Landmark**. In the previous example "above" is a **Relation** while "waterfall" is a **Landmark**. For simplicity, components of structure concepts are omitted from the examples.

GIVER 1: okay ... ehm ... right, you have the start?
FOLLOWER 2: yeah.
GIVER 3: right, below the start do you have ... er like a missionary camp?
FOLLOWER 4: yeah.
GIVER 5: okay, well ... ... if you take it from the start just run ... ... horizontally.
FOLLOWER 6: uh-huh.
GIVER 7: eh to the left for about an inch.
FOLLOWER 8: right.
GIVER 9: and then go down along the side of the missionary camp.
FOLLOWER 10: uh-huh.
GIVER 11: 'til you're about an inch ... above the bottom of the map.
FOLLOWER 12: right.
GIVER 13: then you need to go straight along for about 'til about ... you're about an inch and a half away from the edge of the map.
FOLLOWER 14: the banana tree?
GIVER 15: okay?
FOLLOWER 16: do you have a banana tree?
GIVER 17: i have gorillas that's probably the.
FOLLOWER 18: right okay, i'll go round the banana tree.
GIVER 19: yeah, so you're at where are you now you're at the bottom?
FOLLOWER 20: i'm at the bottom.
GIVER 21: okay, now you ... you need to go parallel to the side of the map.
FOLLOWER 22: uh-huh.
GIVER 23: ehm about ... four inches.
FOLLOWER 24: right.
GIVER 25: so you're just ab-- ... ... y-- ... ... you're now about two inches above the gorillas ... or the banana tree.
FOLLOWER 26: yeah.

Figure 3.19: An example dialog (partial) in the map reading domain

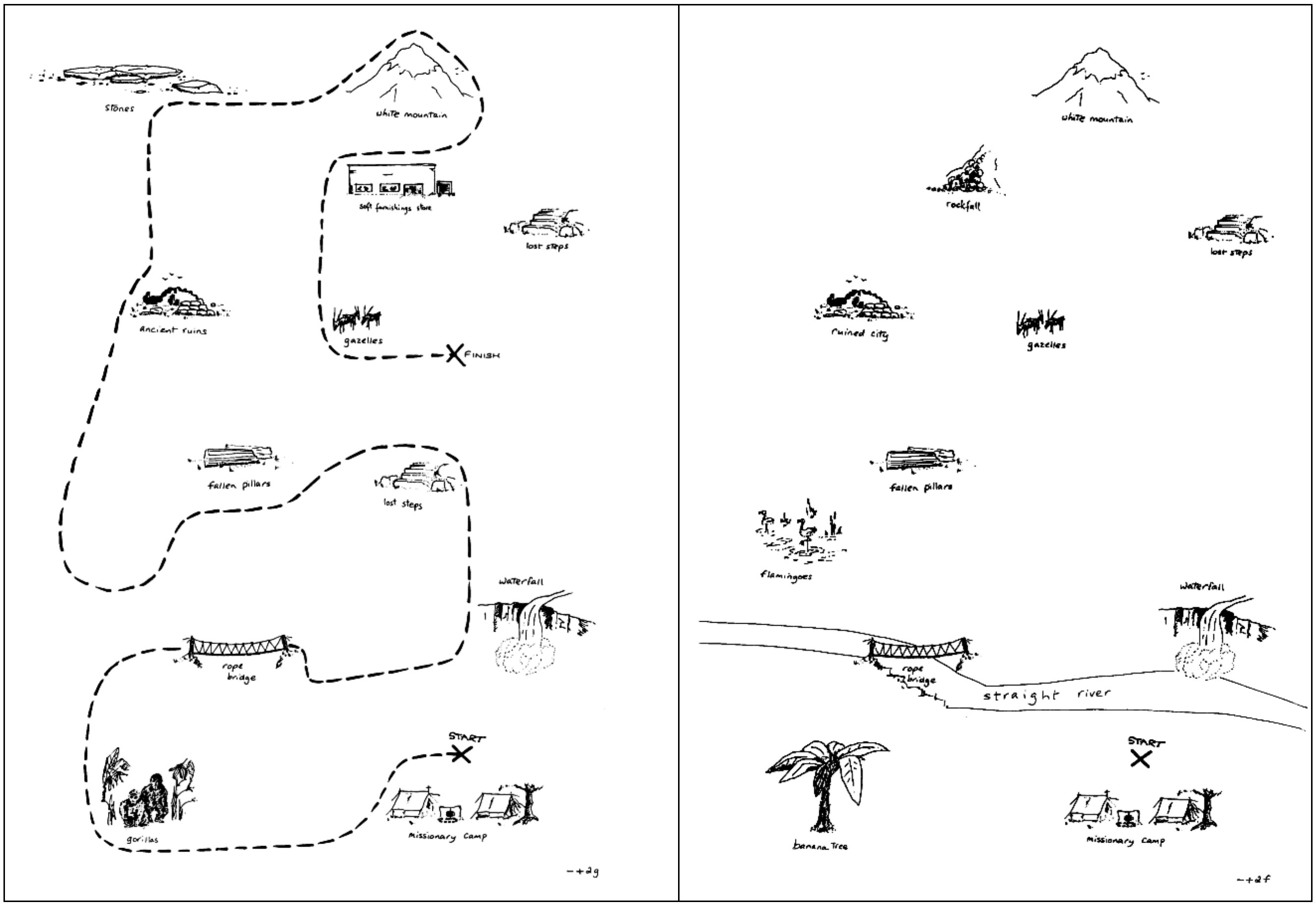


Figure 3.20: A route giver's map and a route follower's map in the HCRC Map Task corpus

There is more than one alternative to describe a route segment. One route giver may use only a **StartLocation** and an **EndLocation** (for instance, go from "A" to "B") to describe the segment while another route giver may use a **StartLocation**, a **Direction**, and a **Distance** (for instance, from "A" go toward "the right" for "3 centimeters") to describe the same segment. Figure 3.21 illustrates two **draw_a_segment** sub-tasks: the first one has only two concepts as shown in the corresponding segment description form while the second one has all five concepts. Since dialog participants usually discuss route segments in order (from start to finish), the start locations of most segments are omitted. The **StartLocation** of the current segment is assumed to have the same value as the **EndLocation** of the previous segment. Omitted concepts are discussed in more detail in Section 3.8.1. After a route follower gathers enough information about a route segment from a route giver, he/she then draws a line that represented the segment on his/her map. This line is an outcome of a ***draw_a_segment*** action.

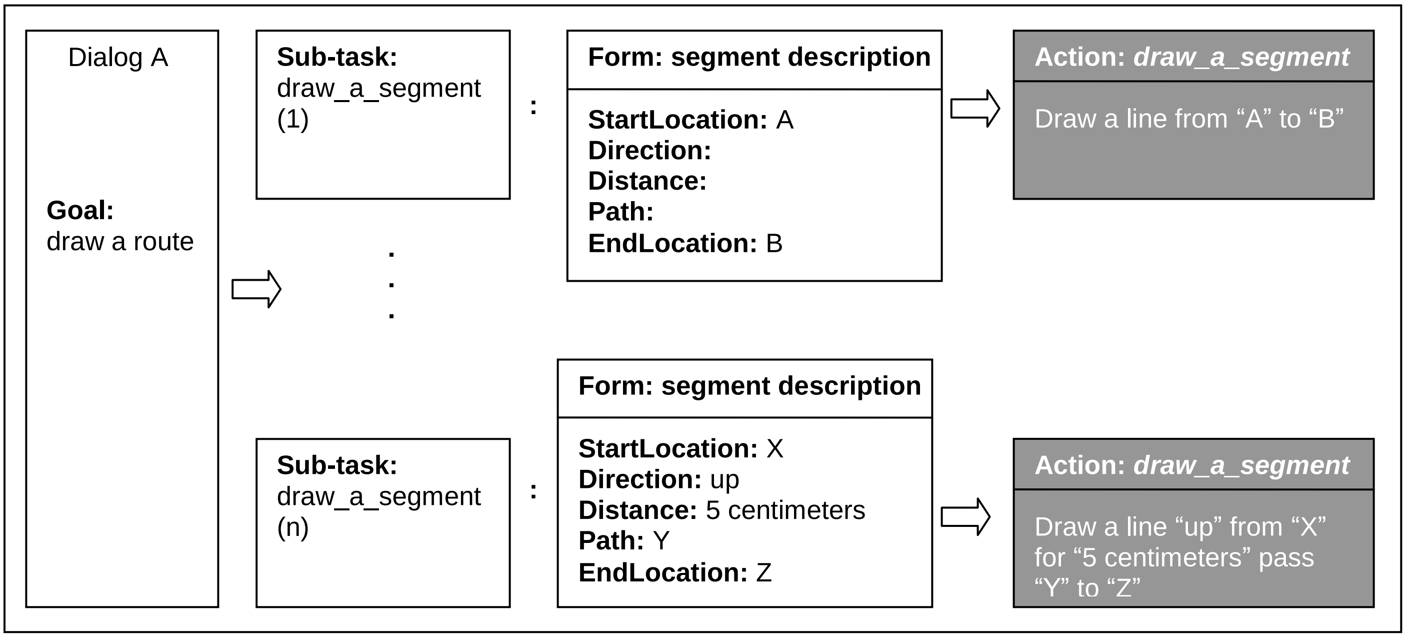


Figure 3.21: A task, sub-tasks and their corresponding forms and actions in the map reading domain

Since there are many ways to describe a route segment, and there is no database backend constraint as in the air travel planning domain and the bus schedule query domain discussed in the previous sections, there is no hard constraint on how a route should be divided into smaller segments . The length and complexity of each segment can be varied depending on a route giver's decision. To identify a boundary between two consecutive **draw_a_segment** sub-tasks, one possibility is to use an occurrence of a ***draw_a_segment*** action since an occurrence of an action usually signals the end of its corresponding sub-task. However, a ***draw_a_segment*** action may not be observed directly from a conversation as there is usually no explicit expression like "I'm drawing a line" that marks the action. Moreover, the outcome of the ***draw_a_segment*** action, which is a route segment drawn on a follower's map, is also not observable from the conversation.

Nevertheless, there is still evidence found within the conversation that indicates the boundaries of a **draw_a_segment** sub-task. I observe a route giver and a route follower usually discuss back and forth on the description of a route segment until the follower has enough information to draw the route segment on the map. A set of concept instances that are used to describe the route segment are repeated within this dialog segment. This set of concept instances is different from the ones used to describe adjacent route segments. Therefore, we can say that a set of concept instances are coherence within a sub-task.

This coherence set of information items can help us identify the boundaries of a **draw_a_segment** sub-task.

The maps of both participants may have some differences, thus some landmarks may be missing from one of the map. The participants have to define the location of a missing landmark before using it in a segment description. This part of a conversation can be considered as *grounding* because it creates mutual understanding on the location of a particular landmark between the participants.

Grounding is a sub-subtask under a **draw_a_segment** sub-task. Figure 3.22 shows a **grounding** sub-subtask and its corresponding form and action. A ***define_a_landmark*** action associates a **LandmarkName** with its **Location** (these are concepts in a **grounding** sub-subtask). The ***define_a_landmark*** action can only be observed when a participant explicitly marks a missing landmark on his/her map. Even though the action may not be observed, the knowledge about the location of the missing landmark is constructed and can be used for future reference by both participants. The ***define_a_landmark*** action occurs when the participants agree on the location of the landmark and storing the information for future use (either by marking the landmark on a map or memorizing it). A **grounding** sub-subtask may occur more than once in one **draw_a_segment** sub-task, or may not occur at all if no landmark needs to be grounded. The **grounding** sub-subtask is an example of an optional sub-task.

In order to know which landmark is missing, the participants check with each other on the existence of a particular landmark on their maps. If the landmark occurs on both maps, they can assume that the location of the landmark is the same and does not have to be explicitly defined. The landmark is implicitly grounded and a ***define_a_landmark*** action also occurs (define the location of the landmark to be the same on both maps). If the landmark is missing from one of the participants' maps, the participant who has that landmark on his/her map describes its location relative to known landmarks. Both types of a ***define_a_landmark*** action (implicit and explicit) are illustrated in Figure 3.23. There is also the case that even though a landmark is missing from one map, the participants do not explicitly ground the landmark. They just avoid using that landmark in a segment description. In this case, a ***define_a_landmark*** action does not occur and the landmark is left ungrounded. Table 3.5 summarizes the structure of a dialog in the map reading domain.

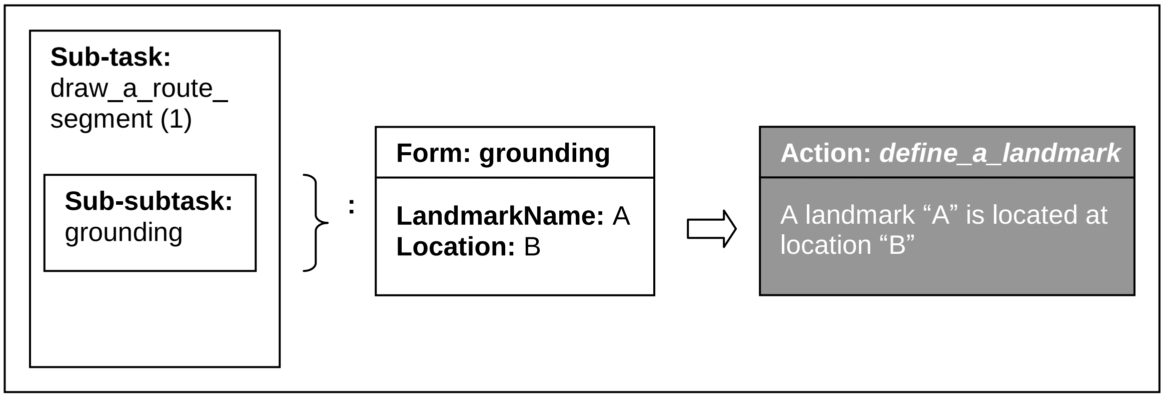
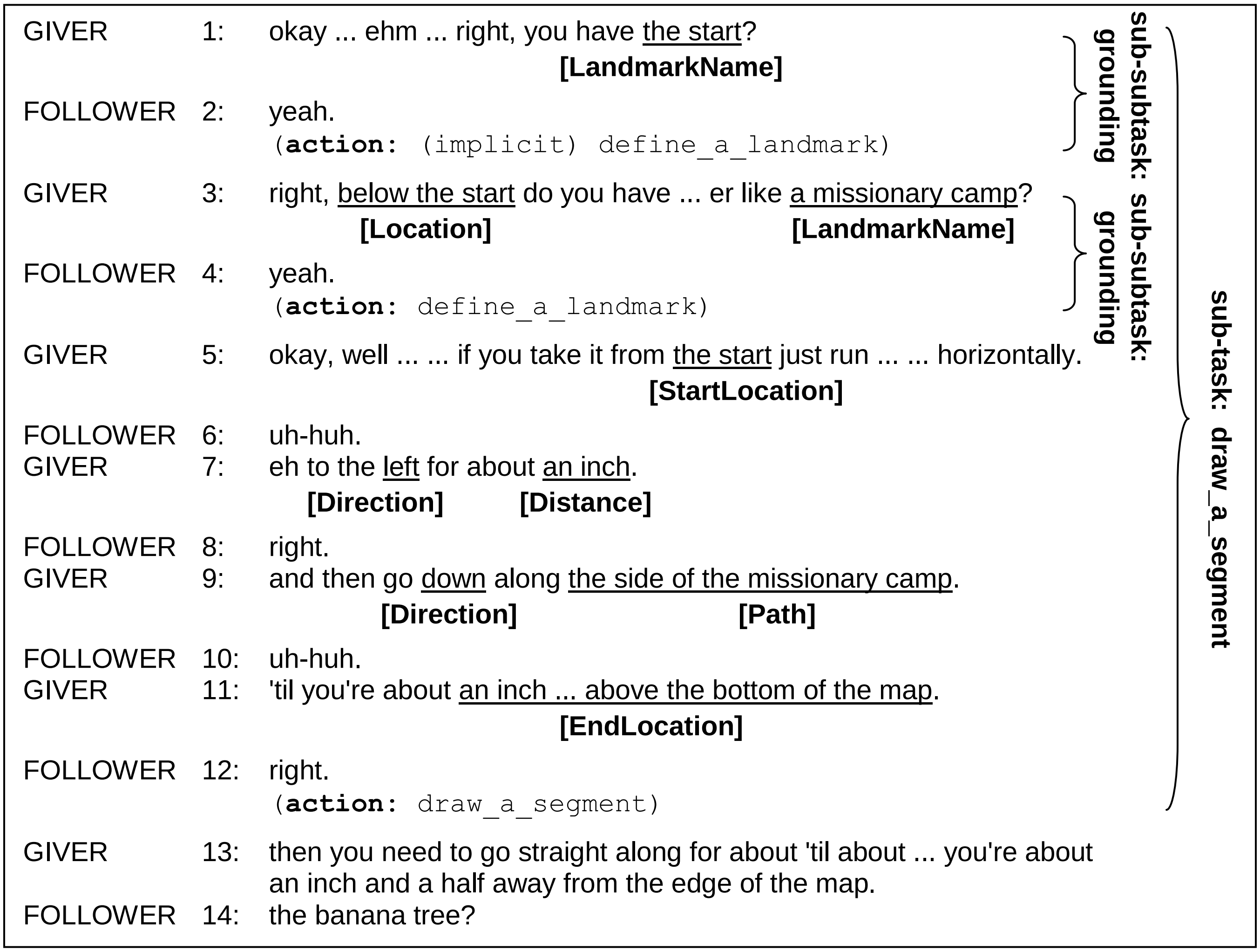

Sub-task: draw_a_route_ segment (1)
Sub-subtask: grounding
:
Form: grounding
LandmarkName: A
Location: B
Action: ***define_a_landmark***
A landmark "A" is located at location "B"

Figure 3.22: A grounding sub-subtask and its associated form and action.

ID	Name	Associated action	Related concepts
Task 1	draw_a_route		
Sub-task 1.1	draw_a_segment	draw_a_segment	StartLocation, Direction, Distance, Path, EndLocation
Sub-subtask 1.1.1	grounding	define_a_landmark	LandmarkName, Location

Table 3.5: Task, sub-tasks and their corresponding actions and concepts in the map reading domain

Figure 3.23 illustrates a dialog structure annotation of the first part of the conversation shown in Figure 3.19. The corresponding maps are presented in Figure 3.20. Similar notions as in Figure 3.12 and Figure 3.16 are used to mark dialog structure components in the dialog. Since the boundaries of a task go beyond the sub-set of the conversation presented in Figure 3.23 (the entire dialog corresponds to a **draw_a_route** task) only sub-task and sub-subtask boundaries are illustrated. Figure 3.24 shows the forms that correspond to all three sub-tasks annotated in Figure 3.23. In the first grounding sub-subtask, "the start" is implicitly grounded since both participants have it on their maps.



GIVER 1: okay ... ehm ... right, you have the start?
[LandmarkName]

FOLLOWER 2: yeah.
(**action:** (implicit) define_a_landmark)

GIVER 3: right, below the start do you have ... er like a missionary camp?
[Location] **[LandmarkName]**

FOLLOWER 4: yeah.
(**action:** define_a_landmark)

GIVER 5: okay, well ... ... if you take it from the start just run ... ... horizontally.
[StartLocation]

FOLLOWER 6: uh-huh.
GIVER 7: eh to the left for about an inch.
[Direction] **[Distance]**

FOLLOWER 8: right.
GIVER 9: and then go down along the side of the missionary camp.
[Direction] **[Path]**

FOLLOWER 10: uh-huh.
GIVER 11: 'til you're about an inch ... above the bottom of the map.
[EndLocation]

FOLLOWER 12: right.
(**action:** draw_a_segment)

GIVER 13: then you need to go straight along for about 'til about ... you're about an inch and a half away from the edge of the map.
FOLLOWER 14: the banana tree?

Figure 3.23: An example of dialog structure annotation in the map reading domain

Form: segment description

Start Location: the start
Direction: left, down
Distance: an inch
Path: the side of the missionary camp
End Location: an inch above the bottom of the map

(a)

Form: Grounding

LandmarkName: the start
Location: (*same on both maps*)

(b)

Form: Grounding

LandmarkName: missionary camp
Location: below the start

(c)

Figure 3.24: The corresponding form of each sub-task in the map reading domain annotated in Figure 3.23

There are other researchers who applied an alternate representation to the same HCRC Map Task corpus for dissimilar purposes. The dialog structure representation proposed by Carletta et al. (1997) describes the compositional structure of a dialog in the map reading domain with a three level hierarchical structure of transaction, conversational game and move. This dialog structure representation is based on the idea of dialog grammar which attempts to model regular patterns in a dialog. A detail discussion about a dialog grammar and dialog structure representations that are based on this idea is provided in Section 2.1.2.3. A transaction in Carletta et al.'s representation, which is a sub-dialog that accomplishes one major step in dialog participants' plan for achieving a task, is quite similar to a sub-task in the form-based dialog structure representation. However, a conversational game and a move are not a decomposition of a task structure as a dialog segment that corresponds to a game or a move is smaller than a step in the task (a sub-set of a conversation that corresponds to one domain action). Both components focus more at the level of intention. A move captures a speaker's intention while a game represents a sequence of intentions (or an initiation-response exchange). Carletta et al.'s dialog structure also does not capture the domain concepts that the participants have to communicate in order to achieve the task goal.

3.5 UAV flight simulation domain

Conversations in the UAV flight simulation domain are taken form the CERTT UAV corpus collected by Cognitive Engineering Research on Team Tasks (CERTT) Laboratory, New Mexico State University (Gorman et al., 2003). A conversation in this domain is an interaction among a pilot, a navigator, and a payload operator, tasked to fly a simulation of an Unmanned Air Vehicle (UAV) on a mission to get photographs of specified targets. The participants had to control the simulated airplane according to various restrictions that were imposed on a route. From the pilot point of view, the conversation in this domain can be considered as a command-and-control task.

Information transfer in a multi-party conversation is more complicated than when there are only two participants in a conversation. In this domain, each participant has different pieces of information that have to be put together in order to perform domain actions. For example, a payload operator has a list of targets while both a payload operator and a navigator know about speed and altitude restrictions. A pilot, on the other hand, does not have any of this information. Therefore, more collaboration, such as volunteering information, is necessary.

PLO -> DEMPC	1:	DEMPC, we have SSTE...
DEMPC -> PLO	2:	Please stand by, PLO.
DEMPC -> all	3:	AVO, I've just sent the route. PLO, go ahead and start naming the targets.
PLO -> DEMPC	4:	Targets are SSTE, Farea, Harea, MSTE, RSTE.
DEMPC -> PLO	5:	Please repeat that again one more time. Farea?
PLO -> DEMPC	6:	Um, it's SSTE, Farea, Harea, MSTE, RSTE, SEN1.
DEMPC -> PLO	7:	That's a roger.
DEMPC -> AVO	8:	AVO, go ahead and proceed to waypoint LVN.
AVO -> DEMPC	9:	We were about 2 miles within there already. I changed course to Harea.
DEMPC -> all	10:	AVO, radius of Harea is five miles. PLO, five miles.
AVO -> DEMPC	11:	Roger.
AVO -> all	12:	That is target area, correct?
PLO -> AVO	13:	Yes. AVO, I need an altitude of greater than 3000.
DEMPC -> AVO	14:	AVO, this is DEMPC. You have to maintain a speed above 50 knots, and stay below 200 knots.
AVO -> all	15:	Roger. Speed changing to 181 altitude climbing to 3300. Currently 71/4 miles out from Harea.
AVO -> DEMPC	16:	Is our next waypoint or target after Harea Farea?
DEMPC -> AVO	17:	Roger.
AVO -> all	18:	How close do we need to get to Farea?
DEMPC -> AVO	19:	Radius is five miles.
AVO -> DEMPC	20:	Roger.
PLO -> all	21:	We got a good photo for Harea.
PLO -> AVO	22:	Uh, AVO recommended altitude lower than 3000, or equal to 3000.
AVO -> all	23:	Changing altitude to 2700, speed 181 151/2 miles from Farea.
DEMPC -> PLO	24:	PLO, this is DEMPC. I need a list again of those targets, another list please. Thank you.
PLO -> DEMPC	25:	DEMPC, this is PLO. I have SSTE, SEN1, and MSTE as well as RSTE.
DEMPC -> PLO	26:	Roger.
AVO -> DEMPC	27:	DEMPC, this is AVO. When you get a chance, can you send me the next waypoint after Farea?
DEMPC -> AVO	28:	Roger, that's a go.
AVO -> PLO	29:	PLO, do I have any speed restrictions right now?
PLO -> AVO	30:	No, not at all.
AVO -> all	31:	Roger, increasing speed to 250.
AVO -> PLO	32:	PLO, is OAK a target?
PLO -> AVO	33:	Negative.
AVO -> PLO	34:	Roger.
DEMPC -> AVO	35:	AVO, this is DEMPC. OAK is our exiting waypoint.
AVO -> DEMPC	36:	Exiting waypoint need to be within five miles, after that is LMR?
DEMPC -> AVO	37:	That is correct. That is the target.
AVO -> DEMPC	38:	Roger.
DEMPC -> AVO	39:	AVO, this is DEMPC. I wanted to say that's not a target. I'm sorry.
AVO -> DEMPC	40:	LMR is not a target.

Figure 3.25: An example conversation (partial) in the UAV flight simulation domain

Since 1) the goal a dialog in this domain is achieved by performing domain actions (controlling a simulated airplane and taking photographs of specified targets), and 2) the dialog participants have to exchange different pieces of information that they have in order to perform these actions through dialog, the characteristics of a dialog in the UAV flight simulation domain match all of the assumptions made by the form-based dialog structure representation. Thus, a dialog in this domain could be modeled by the form-based representation. Detailed analysis of the structure of a dialog in this domain is given below.

A transcript of a conversation in the UAV flight simulation domain is shown in Figure 3.25. Since a conversation in this domain is quite long, only the first part of the conversation is presented. An utterance label indicates both the speaker and the addressees of the utterance. "PLO" represents a payload operator; "DEMPC" represents a navigator, and "AVO" represents a pilot. Since the interaction is among three participants, a speaker may address other two participants together in one utterance as indicated by "ALL". The number that follows the addressees label is an utterance ID. The part of the conversation that is highlighted in blue (utterance 8-21) will be used as an example during dialog structure analysis.

The goal of a conversation in this domain is to take photographs of all specified targets in a given area. There are dependencies among different parts of the conversation as all of the targets are on the same map; therefore, the entire conversation corresponds to only one task, **take_photos** rather than a set of tasks. In order to accomplish the goal, several photos have to be taken, one for each target. A **take_a_photo** sub-task corresponds to a subset of the conversation that contributes toward one ***take_a_photo*** action. The sub-task begins when a particular target is selected as a focused target and ends when a photograph of that target is taken. In order to take a photograph of a target, a plane has to be at a specific distance from the target (an essential radius). Both a **Target** (target name) and a **Radius** are concepts in a target form which associates with a ***take_a_photo*** action. Figure 3.26 shows the decomposition of a task into sub-tasks and their corresponding forms and actions. The first sub-task corresponds to utterance 8 – 21 in Figure 3.25. Only the first part of the second sub-task is included in the example conversation while other sub-tasks are not presented. A detailed annotation of this conversation is given at the end of this section.

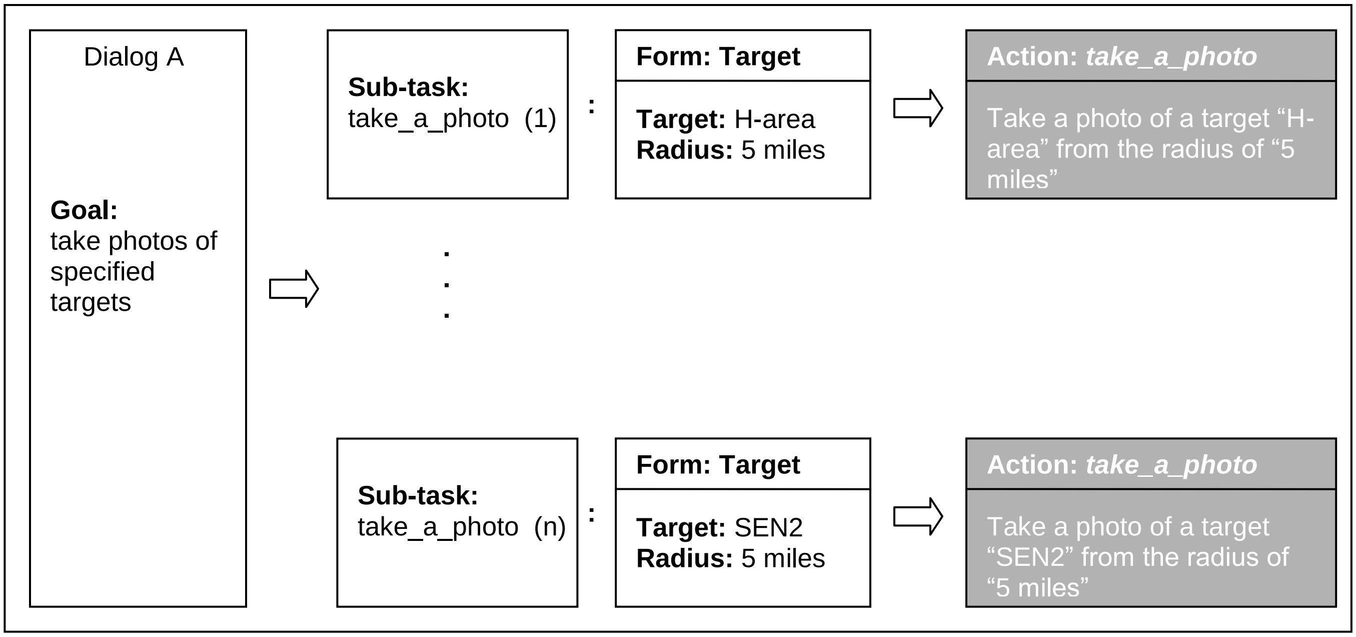


Figure 3.26: A task, sub-tasks and their corresponding forms and actions in the UAV flight simulation domain

In order to take a photograph of a specific target, the participants have to "control a plane" toward the target and "define the type of a landmark" that may appear on a route. These two actions (***control_a_plane*** and ***define_a_landmark***) need to be achieved in order to perform a ***take_a_photo*** action. Therefore, subsets of a conversation that correspond to these actions are sub-subtasks under a **take_a_photo** sub-task. These sub-tasks and actions are described in detail below.

A pilot controls a plane by directing the plane toward a destination, and adjusting its speed and altitude according to the restrictions imposed on a route. A ***control_a_plane*** action involves three concepts that are airplane parameters: **Speed**, **Altitude**, and **Destination**. A **control_a_plane** sub-subtask is a subset of a conversation that discusses these parameters. One or more parameters may be discussed in one **control_a_plane** sub-subtask. The action occurs when the pilot directs the plane toward a new destination (or is ordered to do so) and/or changes the speed or the altitude of the plane. This action changes the position and/or the condition of the plane which the participants may discuss in terms of the new **Speed** and **Altitude**, and a distance from the current destination. A **Distance** is a concept that describes the result of the ***control_a_plane*** action. The ***control_a_plane*** action may occur more than once in one **take_a_photo** sub-task if the airplane parameters have to be adjusted repeatedly with respected to the plane route.

Figure 3.27 shows two **control_a_plane** sub-subtasks that are embedded in the first **take_a_photo** sub-task in Figure 3.26 together with their corresponding forms and actions. The first **control_a_plane** sub-subtask corresponds to utterance 9 in Figure 3.25 and discusses only one concept, **Destination**. The second **control_a_plane** sub-subtask corresponds to utterance 14 – 16 and discusses other two plane's parameters: **Altitude** and **Speed**.

All of the locations mentioned in a conversation are landmarks. There are two types of landmarks: a "target" or a "waypoint" (a point between major points on a route). The participants need to know the type of a landmark in order to plan the route appropriately. A ***define_a_landmark*** action occurs when the type of a particular landmark is discussed in the conversation. The form that associates with this action requires two slots, a **LandmarkName** and its **Type**. The result of the ***define_a_landmark*** action is the knowledge of the type of the landmark that can be used to plan the route. This action might be difficult to observe if no physical action, such as taking note on the landmark type, is performed.

The part of the conversation that associates with a ***define_a_landmark*** action is considered a **grounding** sub-subtask because it creates mutual understanding among the participants about the type of a particular landmark. A **grounding** sub-subtask may occur more than once in one **take_a_photo** sub-task, or may not occur at all if the participants are familiar with all of the landmarks involve in that **take_a_photo** sub-task. The **grounding** sub-subtask is an example of an optional sub-task. A **grounding** sub- subtask and a ***define_a_landmark*** action in this domain are similar to a **grounding** sub- subtask and a ***define_a_landmark*** action in the map reading domain described in Section 3.4. The only difference is the concept that will be associated with a **LandmarkName**. The ***define_a_landmark*** action in the UAV flight simulation domain defines the **Type** of a landmark while the ***define_a_landmark*** action in the map reading domain defines the **Location** of a landmark. Figure 3.27 also shows a **grounding** sub-subtask that is embedded in the first **take_a_photo** sub-task in Figure 3.26 together with its corresponding form and action. This **grounding** sub-subtask corresponds to utterance 12 -13 in Figure 3.25

Unlike a ***define_a_landmark*** action which can be completed instantaneously, a ***control_a_plane*** action may take some time to finish since it might take several minutes before a plane reaches the current destination in the simulation. During that time the participants may occasionally discuss the progress of the plane (the result of the ***control_a_plane*** action) in terms of the **Distance** from the **Destination**, and the current

Speed and **Altitude**. The participants may also plan the next part of the route or discuss about a new target. When this occurs the current **take_a_photo** sub-task is interrupted by the new **take_a_photo** sub-task that corresponds to the new target, and is resumed when the plane approaches the current target and the participants discuss about its corresponding ***take_a_photo*** action. The issues related to the time and physical constraints that cause some actions to be delayed until these constraints are met which are specific to this domain is discussed in more detail in Section 3.8.2. Table 3.6 summarizes the structure of a dialog in the UAV flight simulation domain.

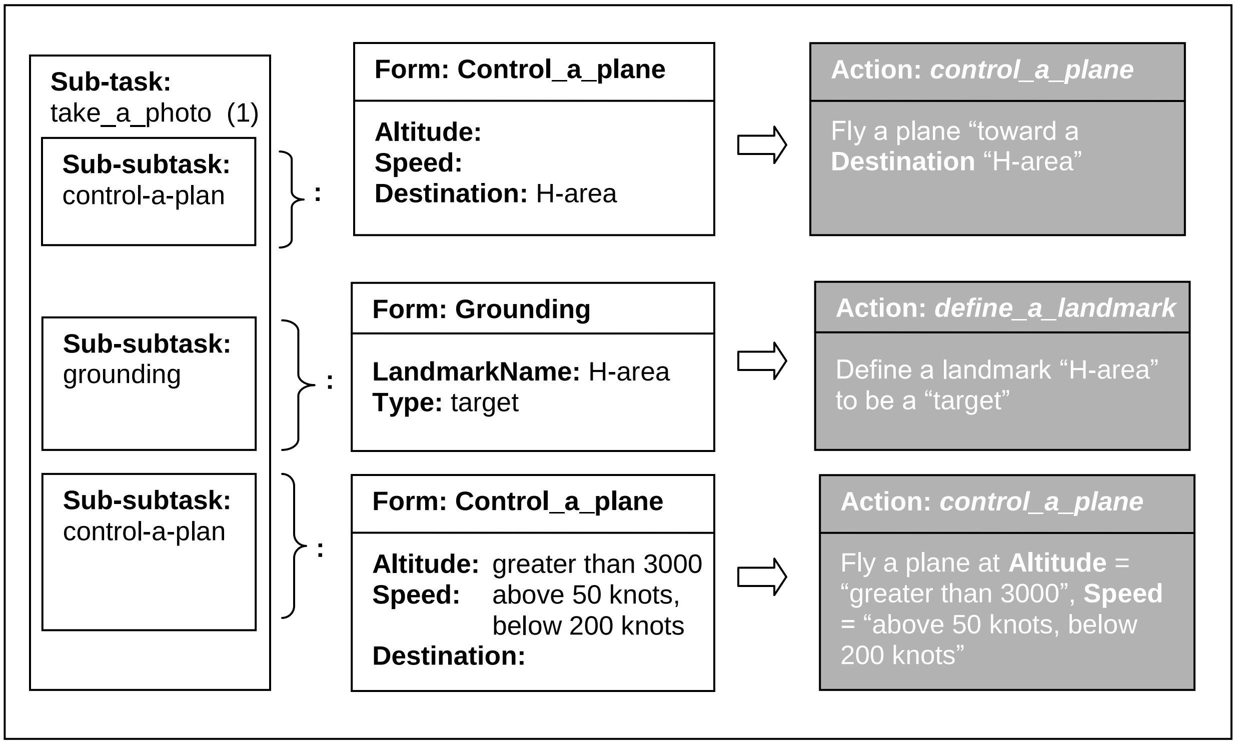

Sub-task: take_a_photo (1)
Sub-subtask: control-a-plan
Sub-subtask: grounding
Sub-subtask: control-a-plan
:
:
:
Form: Control_a_plane
Altitude:
Speed:
Destination: H-area
Form: Grounding
LandmarkName: H-area
Type: target
Form: Control_a_plane
Altitude: greater than 3000
Speed: above 50 knots, below 200 knots
Destination:
Action: control_a_plane
Fly a plane "toward a Destination "H-area"
Action: define_a_landmark
Define a landmark "H-area" to be a "target"
Action: control_a_plane
Fly a plane at Altitude = "greater than 3000", Speed = "above 50 knots, below 200 knots"

Figure 3.27: A grounding sub-subtask and a control_a_plane sub-subtask in the first take_a_photo sub-task

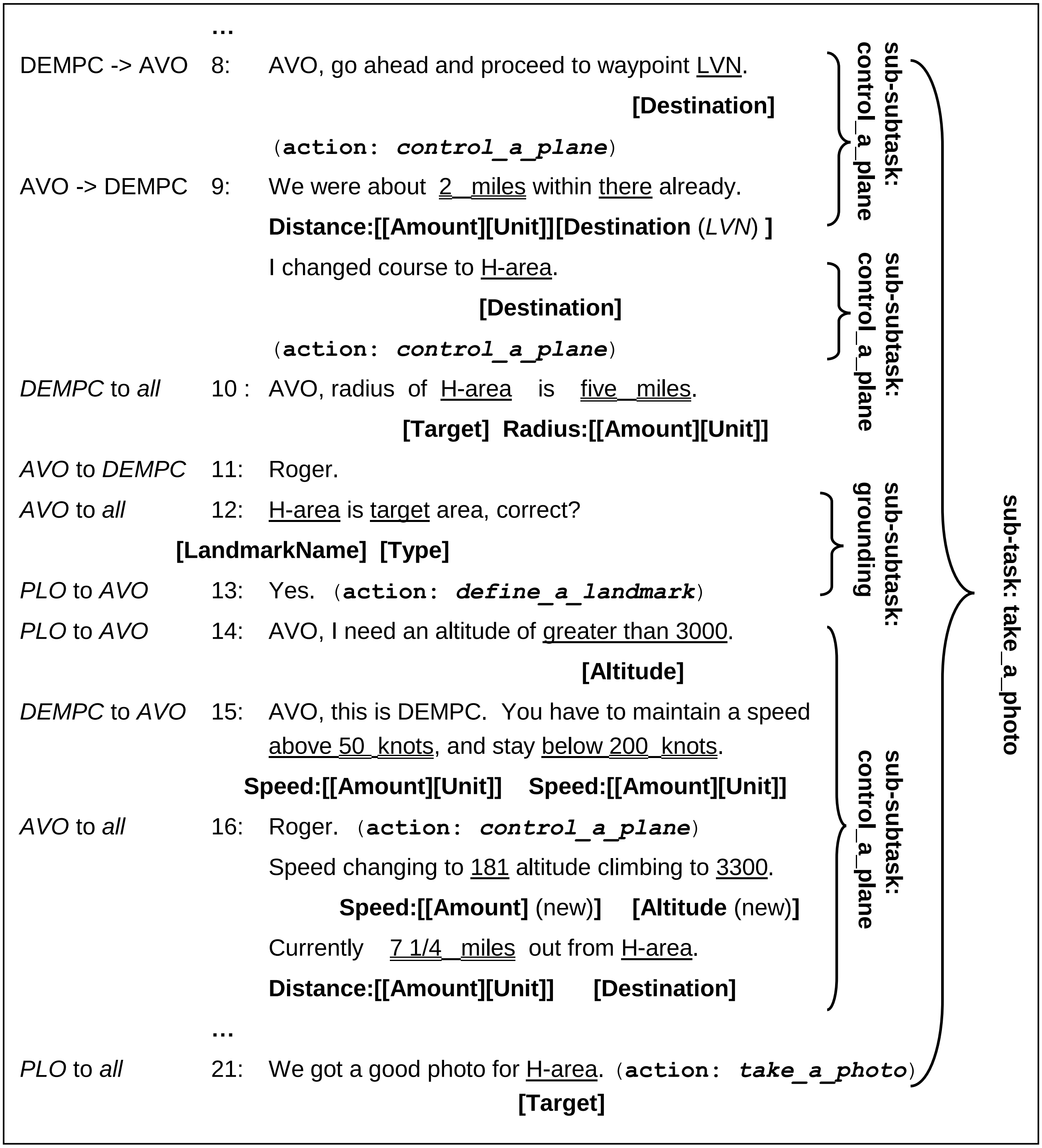
ID	Name	Associated action	Related concepts
Task 1	take_photos		
Sub-task 1.1	take_a_photo	take_a_photo	Target, Radius
Sub-subtask 1.1.1	grounding	define_a_landmark	LandmarkName, Type
Sub-subtask 1.1.2	control_a_plane	control_a_plane	Altitude, Speed, Destination

Table 3.6: Task, sub-tasks and their corresponding actions and concepts in the UAV flight simulation domain

Figure 3.28 illustrates a dialog structure annotation of a subset of a conversation in this domain. This subset of a conversation is the same as the one highlighted in blue (utterance 8-21) in Figure 3.25. For simplicity, the subset of the conversation that discusses the next target while the plane is still heading for the current target is excluded from the analysis. Since the boundaries of a task go beyond the sub-set of the conversation presented in Figure 3.28 (the entire conversation corresponds to a **take_photos** task) only sub-task and sub-subtask boundaries are illustrated. The following notions are used to illustrate the structure of a dialog. Theses notions are similar to the ones used in other dialog structure annotation examples.

- *DEMPC* is a navigator, *AVO* is a pilot and *PLO* is a payload operator.
- The curly bracket on the right shows the boundaries of a sub-task
- An instance of a concept is underlined and the concept name is enclosed in a square bracket underneath it.
- The same word "H-area" is a concept member of 3 different concepts: a **Target**, a **LandmarkName,** and a **Destination** depended on which sub-task it is a part of and its function in that sub-task.
- A **Radius,** a **Distance**, and a **Speed** are structured concepts which compose of two concepts: an **Amount** and a **Unit**. The name of the structured concept is placed on the left-handed side next to its component names.
- (action: …) indicates approximately when an action occurs in a conversation.
- "Roger" means "yes" or "okay".

Figure 3.29 shows the forms that are associated with all five actions in Figure 3.28; a ***take_a_photo*** action (a), a ***define_a_landmark*** action (b) and three ***control_a_plane*** actions (c - e). For simplicity, the labels of the components in structured concepts are excluded. The part of the conversation that plans the new target before a ***take_a_photo*** action occurs is also omitted.



…

DEMPC -> AVO 8: AVO, go ahead and proceed to waypoint LVN.

[Destination]

(**action:** ***control_a_plane***)

AVO -> DEMPC 9: We were about 2 miles within there already.

Distance:[[Amount][Unit]][Destination (*LVN*) **]**

I changed course to H-area.

[Destination]

(**action:** ***control_a_plane***)

DEMPC to *all* 10 : AVO, radius of H-area is five miles.

[Target] Radius:[[Amount][Unit]]

AVO to *DEMPC* 11: Roger.

AVO to *all* 12: H-area is target area, correct?

[LandmarkName] [Type]

PLO to *AVO* 13: Yes. (**action:** ***define_a_landmark***)

PLO to *AVO* 14: AVO, I need an altitude of greater than 3000.

[Altitude]

DEMPC to *AVO* 15: AVO, this is DEMPC. You have to maintain a speed above 50 knots, and stay below 200 knots.

Speed:[[Amount][Unit]] Speed:[[Amount][Unit]]

AVO to *all* 16: Roger. (**action:** ***control_a_plane***)

Speed changing to 181 altitude climbing to 3300.

Speed:[[Amount] (new)**] [Altitude** (new)**]**

Currently 7 1/4 miles out from H-area.

Distance:[[Amount][Unit]] [Destination]

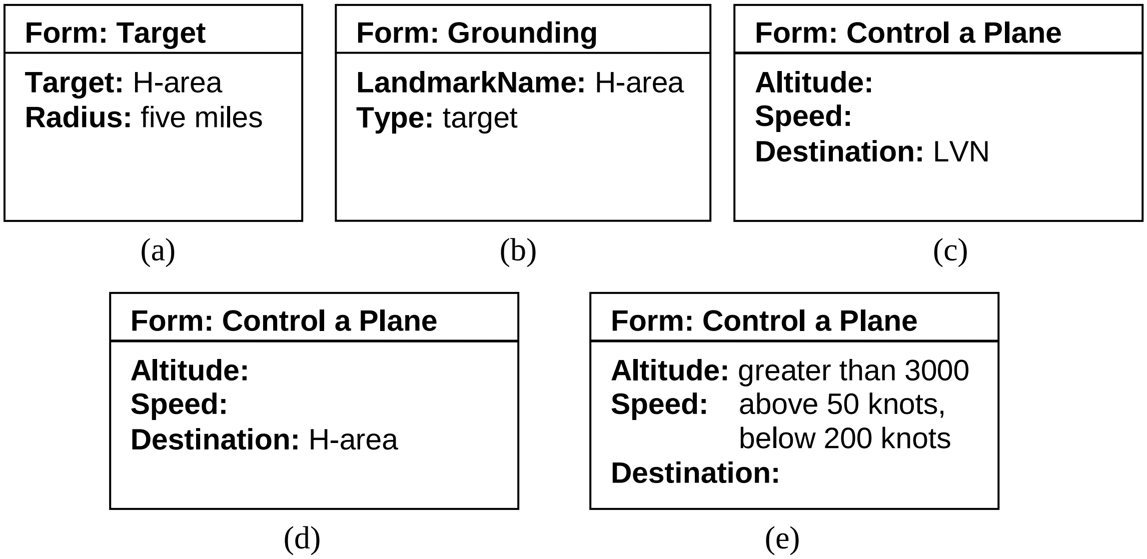
…

PLO to *all* 21: We got a good photo for H-area. (**action:** ***take_a_photo***)

[Target]

sub-subtask: control_a_plane
sub-subtask: control_a_plane
sub-subtask: grounding
sub-subtask: control_a_plane
sub-task: take_a_photo

Figure 3.28: An example of dialog structure annotation in the UAV flight simulation domain



Form: Target

Target: H-area
Radius: five miles

(a)

Form: Grounding

LandmarkName: H-area
Type: target

(b)

Form: Control a Plane

Altitude:
Speed:
Destination: LVN

(c)

Form: Control a Plane

Altitude:
Speed:
Destination: H-area

(d)

Form: Control a Plane

Altitude: greater than 3000
Speed: above 50 knots, below 200 knots
Destination:

(e)

Figure 3.29: The corresponding form of each sub-task in the UAV flight simulation domain annotated in Figure 3.28

3.6 Meeting domain

Conversations in this domain are taken from the CMU CALO[1] Meeting corpus which is the same set of data as the Y2 meeting scenario data described in (Banerjee and Rudnicky, 2006). This corpus contains a collection of multi-party conversations recorded from the meetings in equipment and personnel resource management scenarios. The goal of each scenario is to purchase computers and allocate office spaces to newly hired employees. Each scenario is a series of five meetings; each meeting was being held one week apart. The discussions in subsequence meetings were built up on the decision made in the previous meetings and the progress that was occurred during the one week period. There are three participants in each meeting; each of them takes a role as a manager, a hardware acquisition expert, or a building facility expert. The same set of participants took part in all of the meetings in the sequence. At the end of each meeting, the participants produce a Gantt chart that summarizes all of the decisions made during the meeting. The corpus provides rich information about each meeting including recorded speech from a close-talking microphone and the corresponding transcription, video clips from a long shot video camera and a CAMEO (Camera Assisted Meeting Event Observer), which captures a panoramic view of the meeting, the meeting agenda and notes that all participants took during the meeting. The agenda provides a list of goals

[1] The Cognitive Assistant that Learns and Organizes project http://www.ai.sri.com/project/CALO

that the participants attempted to achieve form the meeting; however, the goals could be adjusted due to dynamic nature of a meeting. The data collection process is described in (Banerjee et al., 2004).

Figure 3.30 shows a transcript of a recorded conversation in the meeting domain. Since a conversation in this domain is quite long, only the first part of the conversation is presented. The nature of this domain is very spontaneous; meeting participants can speak at any time during collaborative discussions. Therefore, a conversation usually contains many interruptions, overlapping speech, and self corrections which cause one speaker turn to be fragmented into several utterances. The marker on the left-handed side indicates the speaker of each utterance (*Hardware Expert* is a hardware acquisition expert; *Building Expert* is a building facility expert). The number that follows the speaker label is an utterance ID. The transcript also includes additional markups for noises and fillers (e.g. /noise/, /uh/, and #begin_background#) made by the participants or occurred in the environment. A dialog structure annotation of the conversation in Figure 3.30 is given at the end of this section. For the purpose of illustration, some examples of dialog structure analysis discussed in this section are taken from other sources.

Purchasing a computer can be viewed as an information-accessing task where a hardware expert searches for computers that match specified criteria from available resources, then presents the result of the search to meeting participants to decide which one they would like to purchase. Similarly, to allocate space to a newly hired employee, a building facility expert searches for available rooms that match specified criteria, then informs the result to the group to decide on the preferred location. Since 1) the conversation goal is achieved by performing domain actions (searching for computers and spaces, and making decisions), and 2) the information that is required to do the search (e.g. computer specification) and to make a decision is discussed through a meeting conversation, the parts of the conversation that discuss the specifications of the desired computer and space, and the parts of the conversations that make the corresponding decisions can be represented by the form-based dialog structure representation similar to the cases of other information-accessing tasks.

Manager 1: we've got a new student coming his name's joe_browning
Manager 2: /uh/ i've got funding for him and i need to get him a computer and some office space
Building Expert 3: /uh/ huh
Manager 4: /uh/ he'll be here on the twenty_first of december
Manager 5: so let's figure out space first where's he going
Building Expert 6: well we should get something
Building Expert 7: /uh/ one space freed on the fourteenth of december
Manager 8: okay
Building Expert 9: /uh/ by paul_smith so i guess if one week is enough for you that should do it
Manager 10: i think that's dependent on a lot of other things
Manager 11: you're the expert on [h(how)] on getting people in
Manager 12: how do we do it
Building Expert 13: well if he just need to i mean the the desk is there if you just need to put the computer if you have the computer one week
Building Expert 14: in advance then
Manager 15: okay
Building Expert 16: i mean one week is more than enough to set up the desk so
Manager 17: okay so if that's we'll plan then
Manager 18: let's get up and write this up
Manager 19: really need to get the facilities people to get us working markers
Manager 20: otherwise it gets pretty hard to do this
Building Expert 21: yeah
Manager 22: alright so what's today the twenty_third
Building Expert 23: yeah twenty_third of november
Manager 24: alright
Manager 25: alright so you said on the twenty_first or rather i said on the twenty_first joe's going to arrive
Building Expert 26: yeah
Manager 27: okay and you said on the fourteenth you'd have space
Building Expert 28: yeah
Manager 29: okay
Manager 30: alright so then we need a computer by then
Manager 31: can we do it
Hardware Expert 32: okay so you need the computer before the fourteenth right
Manager 33: yes
Hardware Expert 34: so that means that we have like about three weeks from now to get
Hardware Expert 35: so you have like any idea what kind of computer do you like to get
Manager 36: well he's going to be working on speech so it needs to be able to do
Manager 37: audio processing for the most part
Hardware Expert 38: yeah and you are looking for like the desktop computer and like what type of operating system do you want him to select there
Manager 39: /uh/ i think for right now we'll work with a desktop /uh/ decent amount of disk space so that he can
Hardware Expert 40: /uh/ huh
Manager 41: you know there's plenty of space to store audio files
…

Figure 3.30: An example conversation in the meeting domain

The goal of purchasing computers and allocating office spaces to newly hired employees discussed above is specific to this meeting scenario. A meeting also has another goal, to delegate a specific work to an appropriate person. This goal is common to a meeting conversation in general. To achieve this goal, a manager assigns a specific work to a person who has the matched expertise. In some cases, a meeting participant may volunteer to do the work. In order to do a work assignment, the description of the work has to be clearly communicated in a conversation along with possible constraints such as the due date. Since the part of the conversation that discusses the work assignment has the characteristics that match all of the assumptions made by the form-based dialog structure representation, this part of conversation can also be modeled by the form-based representation.

Some discussions in a meeting may not contribute directly toward the conversation goals that have been discussed previously. For example, a subset of a conversation highlighted in grey (utterance 9-26) in Figure 3.30, the first part (utterance 9 – 17) contains an additional discussion about the arrangement for a tentatively available room; the second part (utterance 18 – 21) is out of scope of this meeting; the last part (utterance 22 – 26) contains a discussion about the time that a newly hired person would arrive. The discussions in the first part and the last part, while related to the topic of the meeting, do not contribute directly toward the conversation goals. Thus, they are not modeled by the form-based dialog structure representation. This type of sub-dialog is further discussed in Section 3.8.3. Segments of a conversation that do not contribute directly toward the conversation goals are not included in dialog structure analysis described below.

Since each conversation in this domain is a meeting in a sequence of related meetings, discourse structure analysis is done based on the entire sequence rather than on each individual meeting. The ultimate goal of each meeting sequence is to purchase computers and allocate office spaces to newly hired employees. Since all of the meetings in the sequence are based on the same scenario, the entire sequence corresponds to a single task **manage_resource**. In order to achieve the goal, at least two actions are required ***purchase_computer*** and ***reserve_space***. These actions may occur more than once in the sequence if there are several new employees. The **manage_resource** task is decomposed into two types of sub-tasks: **get_computer** and **get_space**. Even though both sub-tasks seem to have a clear goal (i.e. to purchase a computer and to find an office space respectively), they are regarded as sub-tasks instead of tasks because they belong to the same scenario; the decision made in one sub-task may affect the decision made in another sub-task. For instance, a budget constraint may affect the choices of both a computer and an office space. Figure 3.31 shows a decomposition of a **manage_resource** task into two sub-tasks together with the corresponding forms and actions of both sub-tasks. The detail of each form is omitted and will be discussed later on.

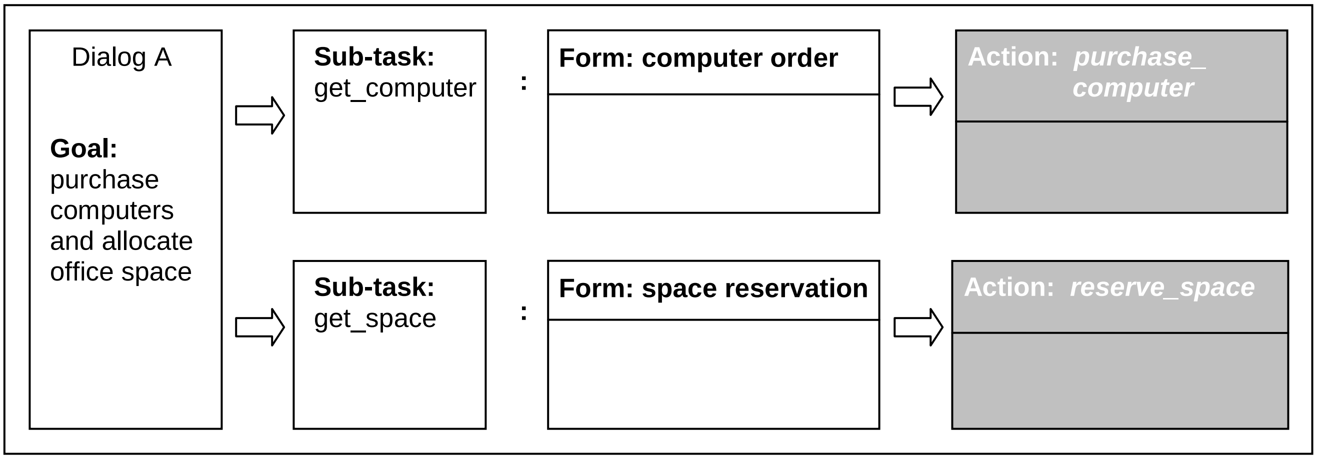
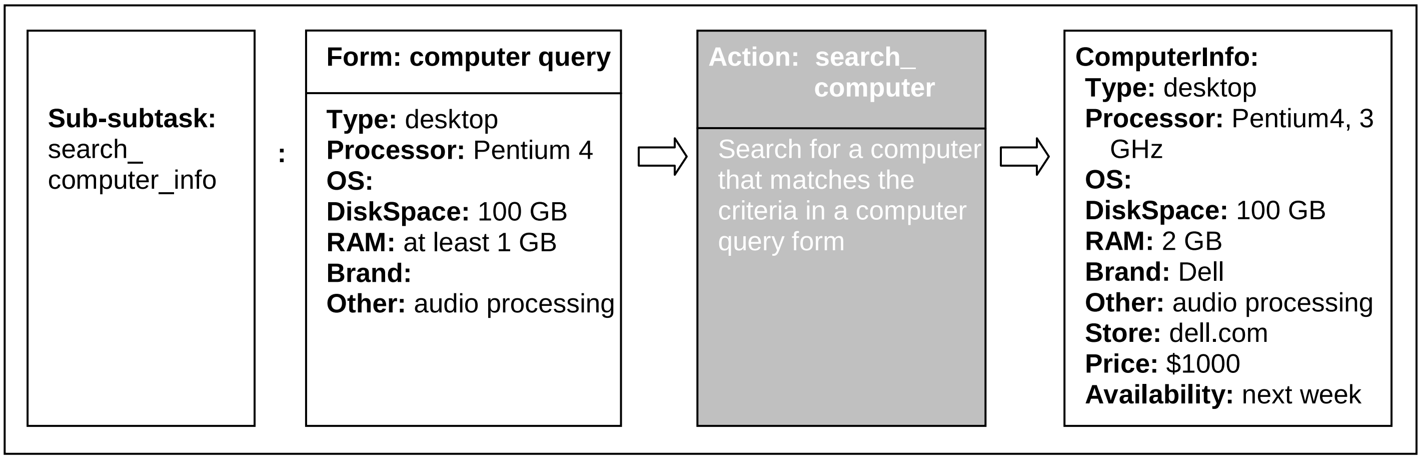


Figure 3.31: A task, sub-tasks and their corresponding forms and actions in the meeting domain

In order to purchase a computer, the meeting participants have to first discuss the specification of the computer that would be suitable for a new employee then check for the price and availability. Based on the information obtained the group decides which computer they will purchase. The computer specification which includes, for example, a **Processor**, an operating system (**OS**), and a **DiskSpace** is a set of concepts in a computer query form. A hardware acquisition expert uses this information to perform a ***search_computer*** action which searches for the computer(s) that matches the criteria from various sources such as internet and a computer store. The result of this action is the information of the computer(s) that has the required specification along with the **Price**, **Store** and **Availability**. This information is regarded as a structured concept, **ComputerInfo**, which contains information about a particular computer. A **search_computer_info** sub-task is the discourse segment that contributes toward the execution of the ***search_computer*** action and discusses its result. The illustration of a **search_computer_info** sub-task, its associated form and action is given in Figure 3.32.



Sub-subtask: search_ computer_info :

Form: computer query

Type: desktop
Processor: Pentium 4
OS:
DiskSpace: 100 GB
RAM: at least 1 GB
Brand:
Other: audio processing

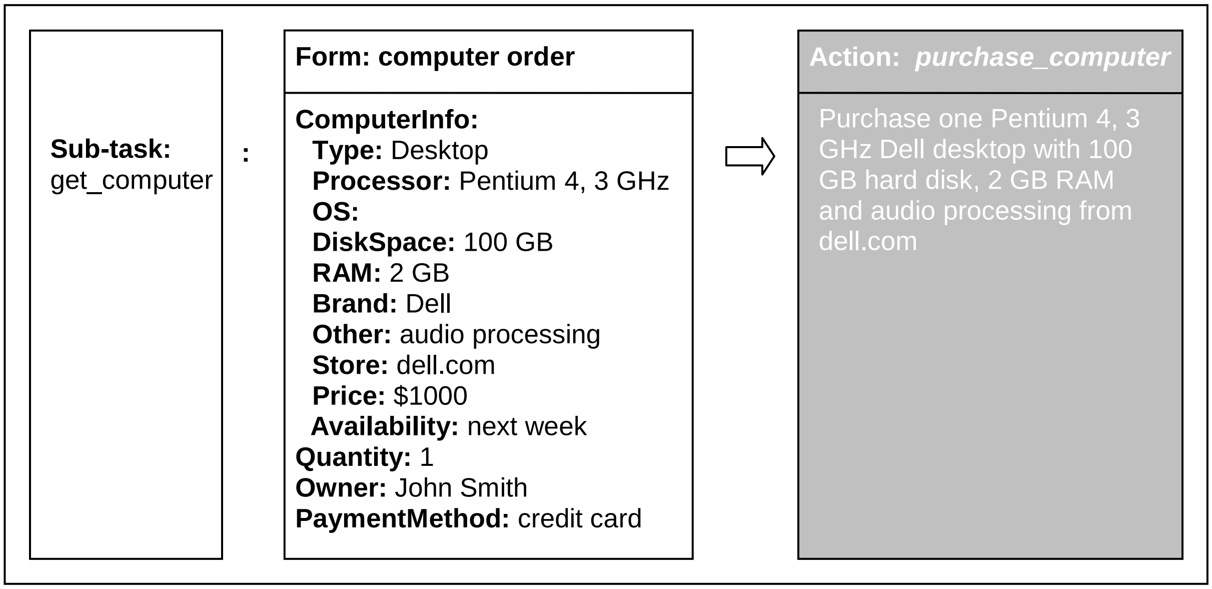
Action: search_ computer

Search for a computer that matches the criteria in a computer query form

ComputerInfo:
Type: desktop
Processor: Pentium4, 3 GHz
OS:
DiskSpace: 100 GB
RAM: 2 GB
Brand: Dell
Other: audio processing
Store: dell.com
Price: $1000
Availability: next week

Figure 3.32: A computer query form and its corresponding action and outcome

After the hardware acquisition expert informs all the meeting participants the possible choices of computers that have the required specification, the group selects the preferred one from the list which makes its corresponding **ComputerInfo** get filled into a computer order form as illustrated in Figure 3.33. The computer order form also contains other concepts that are necessary for purchasing a computer such as an **Owner** and a **PaymentMethod**.



Sub-task: get_computer :

Form: computer order

ComputerInfo:
Type: Desktop
Processor: Pentium 4, 3 GHz
OS:
DiskSpace: 100 GB
RAM: 2 GB
Brand: Dell
Other: audio processing
Store: dell.com
Price: $1000
Availability: next week
Quantity: 1
Owner: John Smith
PaymentMethod: credit card

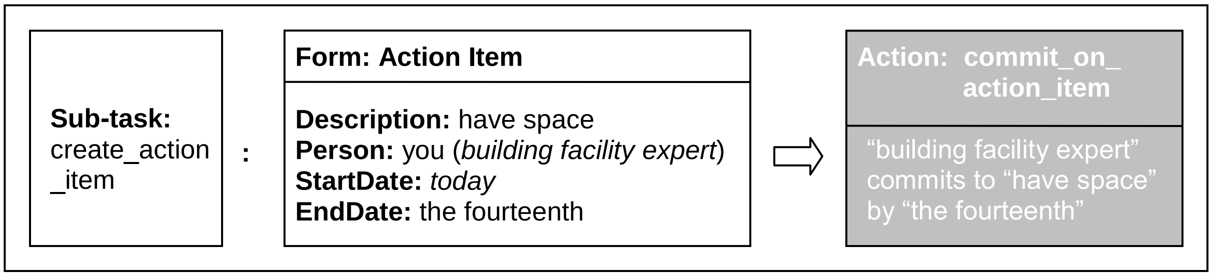
Action: *purchase_computer*

Purchase one Pentium 4, 3 GHz Dell desktop with 100 GB hard disk, 2 GB RAM and audio processing from dell.com

Figure 3.33: A get_computer sub-task and the corresponding form and action

The process for acquiring an office space is quite similar to the process for purchasing a computer. The meeting participants first discuss the criteria of the desired space which include a **Building** and a **RoomLoc**. A building facility expert then uses these criteria, which are concepts in a space query form, to perform a ***search_space*** action which searches for an office space that matches the criteria. The result of this action is the information of the available space(s) along with its **Cost**, **Capacity** and **Availability** time. The result is represented in terms of a structured concept, **SpaceInfo**, which contains information about one particular office space. A **search_space_info** sub-task is the discourse segment that contributes toward the execution of the ***search_space*** action and discusses its result. After the building facility expert informs the participants all of the available spaces, the group then decides on the preferred location which makes its corresponding **SpaceInfo** get filled into a room reservation form. This form also contains other concepts that are necessary for reserving an office space such as an **Owner** and a **MoveInDate**.

All of the task and sub-tasks mentioned earlier corresponds to the ultimate goal and sub-goals of the entire meeting sequence. Since each meeting is a collaborative decision making process, it also produces a set of group decisions. One common form of the decisions is a work assignment, a decision to delegate a specific work to an appropriate person. This type of group decision is also known as an *action item* (or a *to-do item*) (Purver et al., 2006). Segments of a meeting conversation that discuss action items are usually observable. In this meeting domain, the participants have to produce a Gantt chart that summarizes the action items discussed during each meeting. Information in each action item includes a **Description** of the work, a **Person** in charge, a **StartDate** and an **EndDate**. These pieces of information are concepts in an action item form as illustrated in Figure 3.34. A segment of a conversation that discusses an action item is considered a **create_action_item** sub-task and the corresponding action is a ***commit_on_action_item*** action which occurs when the person that is responsible for the action item commits to the work either by accepting the assignment or volunteering to take charge. The **create_action_item** sub-task shown in Figure 3.34 corresponds to utterance 27 – 29 in Figure 3.30. The slot values that are marked in italic are implicit concepts, which have to be inferred from dialog context. Implicit concepts are discussed in more detail below.



Description: have space
Person: you (*building facility expert*)
StartDate: *today*
EndDate: the fourteenth

"building facility expert" commits to "have space" by "the fourteenth"

Figure 3.34: A create_action_item sub-task and the corresponding form and action

Table 3.7 summarizes the structure of a dialog in the meeting domain. While the task and most of the sub-tasks are specific to this equipment and personnel resource management scenario, a **create_action_item** sub-task is quite general in many types of meetings.

ID	Name	Associated action	Related concepts
Task 1	manage_resource		
Sub-task 1.1	get_computer	purchase_computer	ComputerInfo, Quantity, Owner, PaymentMethod
Sub- subtask 1.1.1	search_computer_inf o	search_computer	Type, Processor, DiskSpace, RAM, etc.
Sub-task 1.2	get_space	reserve_space	SpaceInfo, Owner, MoveInDate
Sub- subtask 1.2.1	search_space_info	search_space	Building, RoomLoc
Sub-task 1.3	create_action_item	commit_on_action_item	Description, Person, StartDate, EndDate

Table 3.7: Task, sub-tasks and their corresponding actions and concepts in the meeting domain

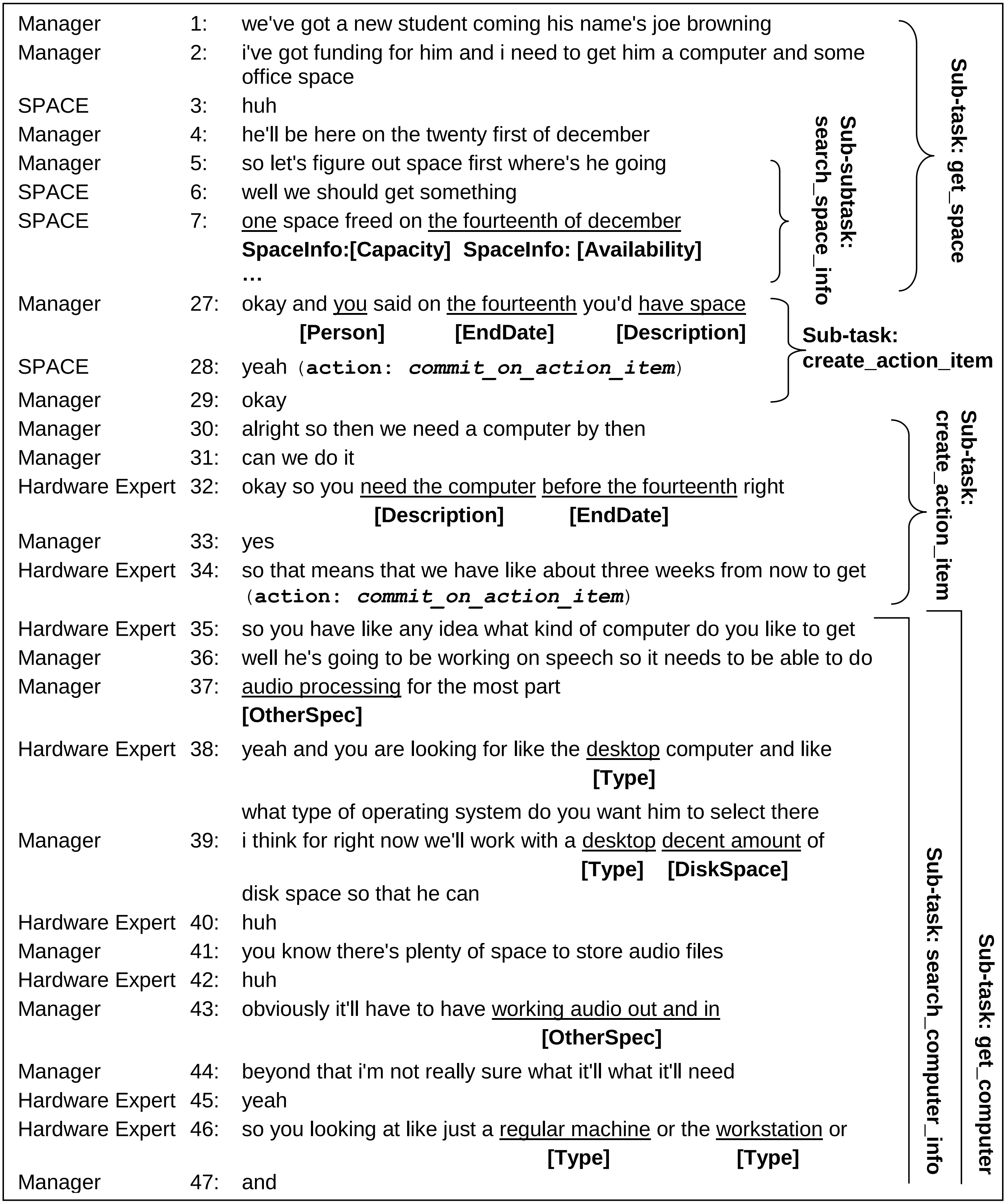
Figure 3.35 illustrates a dialog structure annotation of the first meeting in the sequence of five meetings. Only sub-task and sub-subtask boundaries are illustrated as a **manage_resource** task corresponds to the entire meeting sequence. For illustration purpose, some parts of the conversation are removed from the annotation. The following notions are used to illustrate the structure of a dialog. Theses notions are similar to the ones used in other dialog structure annotation examples.

- *Hardware Expert* is a hardware acquisition expert and *Building Expert* is a building facility expert
- The bracket on the right shows the boundaries of a sub-tasks or a sub-subtask.
- An instance of a concept is underlined and the concept name is enclosed in a square bracket underneath it. For a structured concept, its name is placed on the left-handed side next to its component names.
- (action: …) indicates approximately when an action occurs in a conversation.

One characteristic of the actions in the meeting domain which is different from those in other domains discussed earlier is that some actions are done outside the meeting. In the example conversation, a hardware acquisition expert only provided a tentative and partial result of a ***search_computer*** action using partial information that he/she had at hand. The actual search action was done after the meeting was over and the result of the action, the information of the computer(s) that matches the criteria, was reported in the next meeting (not show in the example). A computer query form, such as the one in Figure 3.36 (a), got filled during the first meeting when all of the participants discussed and agreed on the computer specification. However, since the corresponding ***search_computer*** action was executed later by the hardware acquisition expert, it could not be observed during the meeting. **ComputerInfos** which are the results of the action were reported in the next meeting and one of them was selected and filled into a computer order form. The actual a ***search_space*** action also occurred outside the meeting. A building facility expert only provided a tentative result which he/she had to confirm after the meeting was done. A **search_computer_info** sub-task, a **get_computer** sub-task, a **search_space_info,** sub-task and a **get_space** sub-task may span across multiple meetings. The issues related to fragmented sub-tasks are discussed in more detail in Section 3.8.2.

Some concepts in an action item form may not be observed directly from a conversation and have to be inferred from the context. The participants usually discuss an **EndDate** explicitly but not a **StartDate**. The value of a **StartDate** usually be “today” when it is not mentioned explicitly which is the cases for all action times in Figure 3.36. The **StartDate** of one action item may also depend on the **EndDate** of another action items. For example, an action item “set up a computer” can only start after an action item “get a computer” is finished. Sometimes the participants may not explicitly state the **StartDate** of the succeeding action item as it can be inferred from the **EndDate** of the preceding action item. For an **EndDate**, if its value is omitted from the discussion, the default value should be “next meeting” as shown in Figure 3.36 (g).

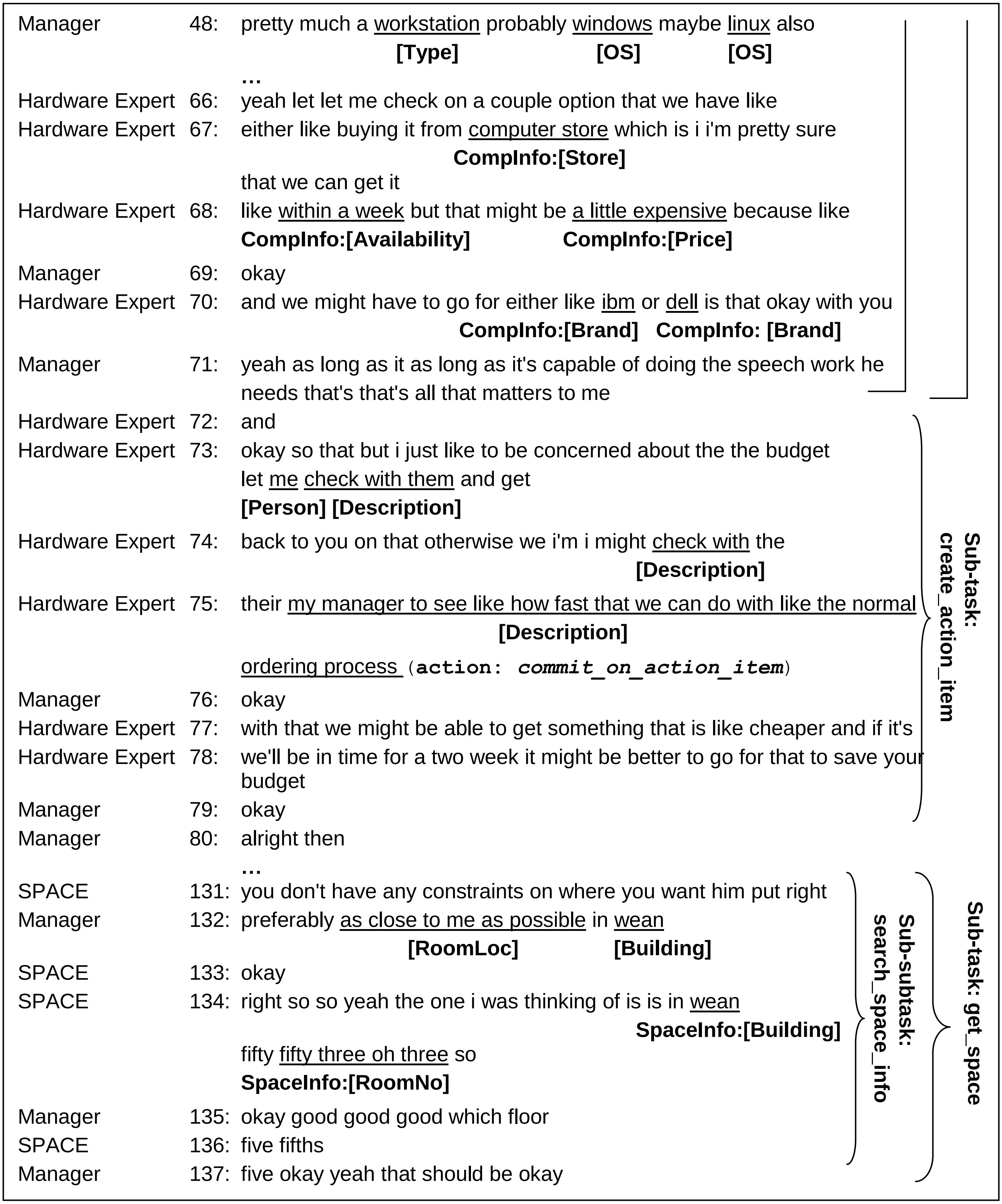
For a **Person** in charge, it is usually referred to by a pronoun such as “you” (when the person is assigned the work) or “me” (when the person volunteers to do the work) as shown in Figure 3.36 (e) and Figure 3.36 (g) respectively. When the value of a **Person** is omitted entirely, it could be inferred from the context using a role detection algorithm which predicts the participant whose expertise matches the description of the action item as the person in charge (Banerjee and Rudnicky, 2006). The implicit concepts are marked in italic in Figure 3.36. Implication of implicit concepts on dialog structure learning is discussed in Section 3.8.1.



Manager	1:	we've got a new student coming his name's joe browning
Manager	2:	i've got funding for him and i need to get him a computer and some office space
SPACE	3:	huh
Manager	4:	he'll be here on the twenty first of december
Manager	5:	so let's figure out space first where's he going
SPACE	6:	well we should get something
SPACE	7:	one space freed on the fourteenth of december
		SpaceInfo:[Capacity] SpaceInfo: [Availability]
		…
Manager	27:	okay and you said on the fourteenth you'd have space
		[Person] [EndDate] [Description]
SPACE	28:	yeah (**action: *commit_on_action_item***)
Manager	29:	okay
Manager	30:	alright so then we need a computer by then
Manager	31:	can we do it
Hardware Expert	32:	okay so you need the computer before the fourteenth right
		[Description] [EndDate]
Manager	33:	yes
Hardware Expert	34:	so that means that we have like about three weeks from now to get (**action: *commit_on_action_item***)
Hardware Expert	35:	so you have like any idea what kind of computer do you like to get
Manager	36:	well he's going to be working on speech so it needs to be able to do
Manager	37:	audio processing for the most part
		[OtherSpec]
Hardware Expert	38:	yeah and you are looking for like the desktop computer and like
		[Type]
		what type of operating system do you want him to select there
Manager	39:	i think for right now we'll work with a desktop decent amount of
		[Type] [DiskSpace]
		disk space so that he can
Hardware Expert	40:	huh
Manager	41:	you know there's plenty of space to store audio files
Hardware Expert	42:	huh
Manager	43:	obviously it'll have to have working audio out and in
		[OtherSpec]
Manager	44:	beyond that i'm not really sure what it'll what it'll need
Hardware Expert	45:	yeah
Hardware Expert	46:	so you looking at like just a regular machine or the workstation or
		[Type] [Type]
Manager	47:	and

Sub-task: get_space
Sub-subtask: search_space_info
Sub-task: create_action_item
Sub-task: create_action_item
Sub-task: search_computer_info
Sub-task: get_computer

Figure 3.35: An example of dialog structure annotation in the meeting domain



Manager 48: pretty much a workstation probably windows maybe linux also
[Type] **[OS]** **[OS]**

…

Hardware Expert 66: yeah let let me check on a couple option that we have like

Hardware Expert 67: either like buying it from computer store which is i i'm pretty sure
CompInfo:[Store]
that we can get it

Hardware Expert 68: like within a week but that might be a little expensive because like
CompInfo:[Availability] **CompInfo:[Price]**

Manager 69: okay

Hardware Expert 70: and we might have to go for either like ibm or dell is that okay with you
CompInfo:[Brand] **CompInfo: [Brand]**

Manager 71: yeah as long as it as long as it's capable of doing the speech work he needs that's that's all that matters to me

Hardware Expert 72: and

Hardware Expert 73: okay so that but i just like to be concerned about the the budget
let me check with them and get
[Person] [Description]

Hardware Expert 74: back to you on that otherwise we i'm i might check with the
[Description]

Hardware Expert 75: their my manager to see like how fast that we can do with like the normal
[Description]
ordering process (**action:** ***commit_on_action_item***)

Manager 76: okay

Hardware Expert 77: with that we might be able to get something that is like cheaper and if it's

Hardware Expert 78: we'll be in time for a two week it might be better to go for that to save your budget

Manager 79: okay

Manager 80: alright then

…

SPACE 131: you don't have any constraints on where you want him put right

Manager 132: preferably as close to me as possible in wean
[RoomLoc] **[Building]**

SPACE 133: okay

SPACE 134: right so so yeah the one i was thinking of is is in wean
SpaceInfo:[Building]
fifty fifty three oh three so
SpaceInfo:[RoomNo]

Manager 135: okay good good good which floor

SPACE 136: five fifths

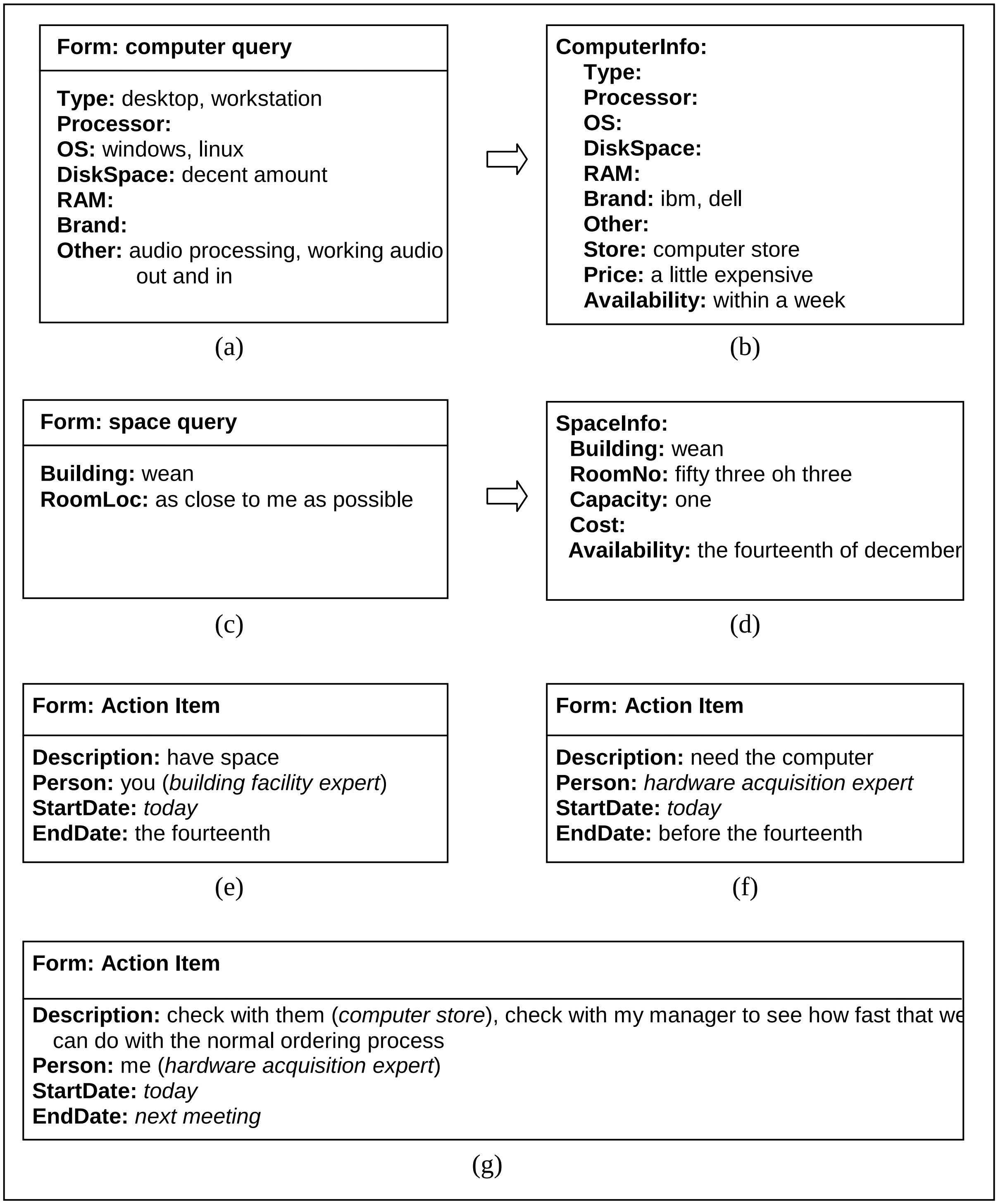
Manager 137: five okay yeah that should be okay

Sub-task: create_action_item

Sub-subtask: search_space_info

Sub-task: get_space

Figure 3.35: An example of dialog structure annotation in the meeting domain (cont.)



Form: computer query

Type: desktop, workstation
Processor:
OS: windows, linux
DiskSpace: decent amount
RAM:
Brand:
Other: audio processing, working audio out and in

⇨

ComputerInfo:
Type:
Processor:
OS:
DiskSpace:
RAM:
Brand: ibm, dell
Other:
Store: computer store
Price: a little expensive
Availability: within a week

(a)

(b)

Form: space query

Building: wean
RoomLoc: as close to me as possible

⇨

SpaceInfo:
Building: wean
RoomNo: fifty three oh three
Capacity: one
Cost:
Availability: the fourteenth of december

(c)

(d)

Form: Action Item

Description: have space
Person: you (*building facility expert*)
StartDate: *today*
EndDate: the fourteenth

(e)

Form: Action Item

Description: need the computer
Person: *hardware acquisition expert*
StartDate: *today*
EndDate: before the fourteenth

(f)

Form: Action Item

Description: check with them (*computer store*), check with my manager to see how fast that we can do with the normal ordering process
Person: me (*hardware acquisition expert*)
StartDate: *today*
EndDate: *next meeting*

(g)

Figure 3.36: All of the forms that correspond to the dialog structure annotation in the meeting domain presented in Figure 3.35

3.7 Tutoring domain

Conversations in this domain are taken from the WHY Human Tutoring corpus (Rosé et al., 2003), a collection of human-human typed dialogs in a physics tutoring domain. Each dialog is an interaction between a tutor and a student during an essay writing process in response to qualitative physics questions. For each problem, the student first types a short essay to answer the question. A good essay should cover all expected propositions and does not contain any misconception. Based on the analysis of the initial essay, the tutor then engages the student in a tutoring dialog to address misconceptions and help the student learns the correct physics concepts. At the end of the conversation, the tutor asks the student to revise the essay. The rewriting process iterates until the tutor satisfies with the essay and then presents the student with a correct essay. Figure 3.37 presents a typed conversation captured from a tutoring session. Since a tutoring dialog for each problem is quite long, only the first part is presented. A typed dialog may include some typos that both participants introduced. Figure 3.37 also shows a question, a student's initial essay before a tutoring conversation starts and a student's revised essay at the end of the conversation.

The ultimate goal of a tutoring session is to help a student learn physics concepts through qualitative explanations. However, only an interactive part (a tutoring dialog) is focused in this analysis. This part is highlighted in blue in Figure 3.37 (sentence 1 – 46). The goal of the tutoring dialog is to help the student improve the quality of an initial essay. However, a tutor cannot simply give away a correct answer but rather helps the student constructs the required knowledge by himself through a knowledge formation process. In this process, the tutor asks the student a series of questions that helps the student learn a specific physics concept such as objects' acceleration or forces acting on objects. If the student gives an incorrect answer to one of the questions, another knowledge formation process will be evoked to help the student develop the underlying physics concept required to answer that question.

In each tutoring dialog, a tutor takes initiative as he/she guides a student through a series of questions that helps the student construct necessary physics knowledge. In order to do so, a list of questions and a correct answer for each question must be specified. This domain information is specific to each physics concept. After analyzing tutoring dialogs in this domain, I found that the proposed form-based dialog structure representation may not be suitable for representing the domain information in the tutoring domain because of the following reasons.

Problem 38. If a lightweight car and a massive truck have a head-on collision, upon which vehicle is the impact force greater? Which vehicle undergoes the greater change in its motion? Defend your answers.

Essay: The lightweight vehicle wll undergo the greater change in its motion because the massive truck has more power and more weight than the lightweight vehicle. The lightweight vehicle might flip over and it would get more damaged the the truck.

Tutor 1: Let us think systematically.Consider any one of the vehicles, what factor determines the change in its motion?
Student 2: velocity
Tutor 3: Velocity describes the motion. What will describe change in motion
Student 4: the weight, i dont know
Tutor 5: What term relates to change in velocity?
Student 6: acceleration
Tutor 7: Right. What determines the acceleration of an object?
Student 8: the speed
Tutor 9: Speed is magnitude of velocity. What is it that causes acceleration?
Student 10: the force on gravity, im not sure
Tutor 11: Yes, force. The force could be due to earth's gravity or due to something else. For example, if an object is sitting on the table(,i.e. it has zero velocity) you want to change its velocity. Tha tis you want it to move. What would you do?
Student 12: I would push the object. so basically what your trying to say is that in this problem we are not sure what kind of force they have so we cant determine which vehicle will have greater change.
Tutor 13: No. I am not saying that. I am trying to show that a force is necessary to cause an acceleration. Nature of force does not matter. In this case the car and the truck collide and each exerts the force of impact on the other. What is the relation between the force experienced by each?
Student 14: they are equal
Tutor 15: Yes, equal in magnitude. How about the dircetion?
…
Tutor 46: Right but some what indeirect. Newton's law uses mass not weight so it should be made clear that less weight means less mass. Secondly, the law gives you acceleration, you should state that larger acceleration means greater change in motion. it is better to be explicit about statements. Can yoiu revise the essay now.

Essay: According to Newton's third law of motion, the two objects interacting to exert a force on each other and the force on each is equal in magnitude and opposite in direction. To determine which vehicle has the greater change in motion during the impact, Newton's second law covers that and it states that an object mass times acceleration is equal to the net force acting on it. Now, less weight means less mass and the larger acceleration means greate change in motion so the lightweight vehilcle will undergo the greater change in its motion due to all of theses arguments.

Tutor 47: Ok. Below is the ideal essay. Indicate when you have read it.
…

Figure 3.37: An example dialog (partial) in the tutoring domain

1. The form-based dialog structure representation does not capture a relation between objects.

   Concepts in the form-based representation are originally designed to capture domain entities, as such location (**CityName**), organization (**RentalCompany**) and dates (**StartDate**), rather than relations between entities. However, in the tutoring domain, relations between entities are also essential for determining the correctness of each student answer. For example, in sentence 14, "equal" is a relation between the force experienced by the car and the force experienced by the truck and was inquired by the tutor in the previous sentence.

2. The form-based dialog structure representation models types of information rather than specific values.

   As described in Section 3.1, each form-based dialog structure component has two aspects: type and instance. A type is an abstraction of similar information items (for example, **CityName)** while an instance is a specific value of an information item (for example, "pittsburgh"). The form-based dialog structure representation models pieces of domain information by their types rather than instances. A slot in a form corresponds to a concept type; therefore, it represents any instance of that concept not just one specific instance. For example, a slot **CityName** represents any city name not only one specific city name "pittsburgh". However, in the tutoring domain, some domain entities and relations do not require the types. For example, physics terms such as "velocity" and "force" must be represented by the terms themselves. There is no notion of type abstraction for such terms. Moreover, for each question, certain values of domain entities and relations must be specified in the answer to be considered as correct. Hence, the abstract representations of forms and slots may not be suitable for representing domain information in this domain.

3. The characteristic of a tutoring dialog does not match the assumption made by the form-based dialog structure representation.

   The form-based dialog structure representation is developed based on the assumption that a conversation goal is achieved through the execution of domain actions and dialog participants gather pieces of information required to perform these actions through dialog. However, in a tutoring dialog, the conversation goal, which is to help a student improve the quality of an initial essay, is achieved through a knowledge formation process rather than through the execution of particular domain actions. Even though, a tutor asks the student a series of questions in the knowledge formation process, both the student and the tutor do not gather pieces of information to perform any domain action through these questions. Each question helps the student formulate the underlying physics concept while helps the tutor discover gaps in the student's knowledge. By answering the questions, the student gradually learns the target physics concept as the dialog progresses.

Furthermore, a tutoring dialog is quite complex since the structure of a dialog can change dynamically depended on the student's answers. For example, the tutor may change the teaching strategy if the student seems to have a lot of problem understanding the target physics concept. The target concept may also need to be changed to address a more crucial misconception. The form-based dialog structure representation, which models a dialog with a rather simple representation, is not suitable for modeling a complex dialog that has a dynamic structure as discussed at the beginning of this chapter.

To resolve the first problem, the form-based representation also needs to model relations between domain entities. Even though a concept may be extended to represent a relation, a representation that is designed specifically for describing complex relations between objects should be more effective. Rosé and Hall (2004) used a predicate language representation to describe complex properties of objects and their relationships in the same physics tutoring domain. The predicate language representation also facilitates deep semantic analysis of student utterances which is essential for determining the correctness of student answers and providing appropriate feedback. For the second problem, a *constant concept* can be added to the form-based dialog structure to represent a domain entity that does not require type abstraction. For example, physics terms such as "force" can be modeled as a constant concept **Force**. Solving the third problem requires modification to the assumption made by the form-based dialog structure representation.

Since extensive modification is required in order to represent a tutoring dialog with the form-based dialog structure representation, we can conclude that the form-based dialog structure representation is not suitable for representing domain information required to build a dialog system in the tutoring domain.

3.8 Difficulties in applying the form-based dialog structure representation

It has been shown in the previous sections that the proposed form-based dialog structure representation can be used to model domain-specific information in various types of task-oriented domains except for the tutoring domain as discussed in Section 3.7. However, among the five domains that it can be applied, air travel planning (information-accessing), bus schedule inquiry (information-accessing), map reading (problem-solving), UAV flight simulation (command-and-control), and meeting, there are some cases that are quite difficult to model with the proposed form-based representation. These complicated cases are implicit concept values, fragmented sub-tasks, and non-task dialog segments. Detailed discussions of these cases are provided in Section 3.8.1 - 3.8.3 respectively. Section 3.8.4 discusses potential difficulties that one might encounter when applying the form-based dialog structure representation to other task-oriented domains besides the ones discussed in this chapter.

3.8.1 Implicit concept values

In some dialogs, dialog participants may not express the values of some concepts explicitly. In some cases, indirect expressions are used while in other cases the values are omitted entirely. The former are considered *indirect concepts* while the latter are considered *omitted concepts*. The following table shows examples of indirect concept values from various task-oriented domains.

Indirect concept value	**Concept type**	**Domain**
“I have a five forty flight”	**ArriveTime**	bus schedule inquiry
“the top of it”	**Location**	map reading
“as close to me as possible”	**RoomLoc**	meeting
“before the new person arrive”	**EndDate**	meeting

Table 3.8: Examples of indirect concept values

Indirect concept values are rarely used in information-accessing domains (air travel domain and bus schedule inquiry domain). A client in these domains usually expresses the criteria for the desired flight or bus clearly, so that an agent does not misunderstand the information. The first example in Table 3.8 occurred when a client used the departure time of his flight as an implicit arrival time of the desired bus instead of the actual time that he would like to arrive at the airport. In the UAV flight simulation domain, due to the characteristic of a command-and-control task, indirect concept values are also rarely used. Moreover, some concept values are specified in a scenario that is not shared among dialog participants; hence, they have to explicitly communicate these concepts. In the map reading domain indirect concept values, mostly pronouns that refer to landmarks, are used occasionally.

However, indirect concept values are used quite often in the meeting domain. Meeting participants are usually familiar with each other and share some common domain knowledge; therefore, some concept values do not have to be mentioned explicitly in a conversation. For example, "as close to me as possible" was used as an implicit value for a **RoomLoc** in a space query from shown in Figure 3.36 (c) instead of an explicit value such as "close to the room 5324". Furthermore, a meeting conversation is not self-contained. The participants often refer to the information that occurs outside the meeting. Examples of this information are the e-mail sent out before the meeting and the information discussed in the previous meeting. Within the same meeting, there are some discussions that do not contribute directly toward any domain action by themselves but have influence on the values of the concepts. The participants may refer to the information in those discussions as concept values instead of using explicit expressions as shown in the last example in Table 3.8. The exact **EndDate** of an action item "get office space" has to be inferred from the arrival date of a newly hired employee which was discussed in another part of the meeting. Many pronouns are also used as indirect concept values in the meeting domain as discussed in Section 3.6.

Indirect concept values are ambiguous; for instance, a pronoun can refer to any kind of noun. To determine an actual value of an indirect concept, additional information besides the indirect expression itself is required. For some indirect concept values, contextual information from within a dialog is sufficient for determining their actual values while other indirect concept values require additional domain and world knowledge that are not presented in a dialog in order to interpret the indirect expressions. Examples of the first type of indirect concept value are pronouns and indirect temporal expressions (e.g. "before the new person arrive"). For this type of indirect concept value, additional understanding processes, such as anaphora resolution and temporal reasoning, can be used to infer the actual concept value from dialog context. Examples of the second type of indirect concept value are the first and the third examples in Table 3.8. To determine the exact value of the **ArriveTime** in the first example, the relation between the departure time of the flight (an indirect value) and the time that a client needs to arrive at the airport (an exact value) has to be provided. For the third example, the location of a manager's office is needed in order to resolve the exact value of the

RoomLoc. These pieces of information are shared among dialog participants, so they are not mentioned in a dialog. An understanding module that interprets this type of indirect concept value has to model the necessary information separately such as in a knowledgebase.

An actual value of an indirect concept is necessary in order to understand a dialog. However, for domain knowledge acquisition, as in the case of this thesis, an actual value of an indirect concept may not be as crucial. In this thesis, the components of the form-based dialog structure representation, such as a list of concepts, are inferred from a set of in-domain conversations. Therefore, if only a few instances of a particular concept type have indirect values, a machine learning algorithm can still utilize evidences from other instances to identify that concept. On the other hand, if many instances of a particular concept type have indirect values, it might be difficult to identify that concept without determining the actual values of those indirect concept instances.

Some concepts are omitted from a conversation even though they are required in order to perform domain actions. These concepts are commonly known by all of the participants, so they do not have to mention them again in the conversation. The participants infer the omitted concepts from the following information:

- Common knowledge about the world or the domain

  In the bus schedule query domain, the date of travel is usually omitted and is assumed to be "today". In the map reading domain, a location of a landmark may be omitted from a grounding form. If both participants discover that they have the same landmark name on their maps, they can assume that the location of the landmark is the same on both maps and does not have to be explicitly defined. In the meeting domain, a StartDate of an action item is rarely specified and is assumed to be "today".

- Dependencies with other concepts

  In the air travel planning domain, many concepts in a car query form and a hotel query form are not explicitly specified. These pieces of information can be inferred from related concepts in a flight reservation form since there are a lot of dependencies among a flight reservation, a car reservation and a hotel reservation as discussed in Section 3.2. In the map reading domain, since a route giver and a route follower usually discuss route segments in order (from start to finish), the start locations of most segments are omitted. They are assumed to have the same values as the end locations of the previous segments. In the meeting domain, meeting participants may not explicitly state the **StartDate** of the succeeding action item as it can be inferred from the **EndDate** of the preceding action item.

Similar to the case of indirect concepts, omitted concepts occur more often in the meeting domain than in the other domains due to the characteristic of the meeting domain discussed above. To process a dialog that contains an omitted concept, additional knowledge, such as a user profile and a default concept value, is required in order to determine the value of the omitted concept. For domain knowledge acquisition, an omitted concept is more problematic than an indirect concept value. If many instances of a particular concept are omitted, it may not be possible to identify that concept using only information from in-domain conversations.

3.8.2 Fragmented sub-tasks

When dialog participants divide a complicated task into a series of sub-tasks, they usually pursue the sub-tasks one at a time, completing one sub-task first before moving on to another one. For example, in the air travel planning domain, a travel agent and a client usually discuss the first leg of an itinerary until the client selects the desired flight before moving on to discuss the second leg. Nevertheless, in some dialogs, a sub-task may be interrupted by another sub-task before dialog participants can gather enough information to execute the associated action or finish the discussion about the outcome of the action. The participants may resume the interrupted sub-task later in the conversation. When a sub-task is interrupted we can say that it is fragmented since the entire sub-task is not represented by one continuous segment of conversation.

Fragmented sub-tasks occur more often in some domains than other domains due to different characteristics of task-oriented domains. In the bus schedule inquiry domain, each task is quite simple and does not have to be decomposed into sub-tasks. Hence, there is no fragmented sub-task problem. In the map reading domain, since drawing a route on a map has a sequential characteristic, the **draw_a_segment** sub-tasks are discussed in order from the start of the route to the end of the route. Only one route segment is discussed at a time until both a route giver and a route follower agree on the segment description and the route segment is drawn on the follower's map. Then they move on to the next segment of the route. Therefore, there is also no fragmented sub-task problem in the map reading domain. If there is a serious understanding problem on a segment description, the participants may restart that sub-task again. In the air travel domain, a make reservation part of a **reserve_flight** sub-task may be interrupted by a **reserve_car** subtask or a **reserve_hotel** sub-task as shown in Figure 3.12. The

reserve_flight sub-task is usually resumed at the end of the conversation. Since a plane ticket reservation is mandatory in this domain while car and hotel reservations are optional, a travel agent may want to focus on the purchase options of the ticket when all other optional discussions about car and hotel have been done.

Some actions in the UAV flight simulation domain cannot be executed or completed immediately after all required concepts have been gathered. Physical and time constraints cause these actions to be delayed until all constraints are met. For example, a ***take_a_photo*** action cannot take place until a plane reaches a target. The interval between the time that all of the required slots have been filled, and the time that all physical and time constraints are met and the action is ready to be executed (or completed) may take several minutes. During this time dialog participants may discuss in preparation for a new sub-task which cause the current sub-task to be fragmented. For example, if the participants discuss about a new target while waiting for the plane to reach the current target in order to execute a ***take_a_photo*** action, the current **take_a_photo** sub-task is interrupted by another **take_a_photo** sub-task. A **control_a_plane** sub-task may also be fragmented if the participants start planning the next destination while the plane is still heading toward the current destination.

In the meeting domain, some actions are executed outside the meeting. This makes the part of a sub-task that fills the corresponding form and the part of a sub-task that discusses the outcome of an action take place in different meetings. For instance, a space query form may be filled during the first meeting when the meeting participants discuss and agree on the criteria of a desired office space. The ***search_space*** action actually occurs after the meeting is done and its result, the office spaces that match the criteria, is discussed in the next meeting. Since a sub-task can span across meetings, it is likely to be interrupted by other discussions that occur in those meetings. Within the same meeting, the participants may revisit the sub-task in order to add or change the values of the slots in the corresponding form as the form is not yet be executed. The dialog structure annotation in Figure 3.35 shows that a **get_space** sub-task and a **search_space_info** sub-task are revisited again at the end of the conversation as the participants specified additional criteria for a desired office space. Other sub-tasks that may span across multiple meetings are a **search_computer_info** sub-task, a **get_computer** sub-task. The characteristics of the meeting domain make the sub-tasks more likely to be fragmented than other domains.

Depending on how the sub-tasks are fragmented and how often they occur, fragmented sub-task instances could make it more difficult to learn the form and slots that are associated with that type of sub-task. In most of the cases discussed above a sub-task gets interrupted after all of the concepts that are necessary for performing an action are discussed but before the action can be executed or completed. This type of fragmented sub-task is less problematic as the slots in the corresponding form are presented in one continuous dialog segment. However, when a sub-task is interrupted before all of the necessary slots in the corresponding form are filled, as in the case of sub-task revisiting in the meeting domain, it may be difficult to identify all of the relevant slots for that type of sub-task if many of it instances have this kind of fragmentation.

For other kinds of dialog structure analysis, fragmented sub-tasks can become more problematic. For example, interrupting sub-tasks make it more difficult to recognizing participants' focus of attention at run time as there are multiple active sub-tasks that could receive the focus of attention. Interruption in a form-based dialog system can be handled by a stack as described in (Constantinides et al., 1998); nevertheless, this issue is not the main focus of this thesis.

3.8.3 Non-task dialog segments

The form-based dialog structure representation focuses on modeling domain-specific information necessary for performing domain actions that contribute directly toward a conversation goal. Since human-human conversations are rich in nature, there might be some segments of a dialog that are not directly related to the domain actions and the dialog goal. These *non-task* dialog segments cannot be explained with the proposed dialog structure representation. Examples of non-task segments are a client's comment on the price of a car rental in the air travel planning domain (utterance 16 -19 in Figure 3.38), and a discussion about a direction from a bus stop to a client's actual destination in the bus schedule inquiry domain (utterance 25 – 30 in Figure 3.38). A non-task segment may have some influence on the interpretation of an indirect concept value as discussed in Section 3.8.1. However, the content of the non-task segment itself does not contain domain-specific information or have any effect on the form.

…
AGENT 13: WITH THE DISCOUNT NUMBER FROM C.M.U. IT COMES TO FIFTY FOUR NINETY NINE FOR HERTZ
CLIENT 14: FIFTY FOUR NINETY NINE
AGENT 15: /MM/ /UM/
CLIENT 16: OKAY THAT SOUNDS ABOUT RIGHT PRICES ARE GOING UP
AGENT 17: THEY ARE HERTZ IS A LITTLE BIT MORE EXPENSIVE THAN SOME OF THE OTHER COMPANIES
CLIENT 18: BUT YOU GET TO DRIVE FORDS
AGENT 19: /LG/ OKAY
…

Figure 3.38: An example of a non-task dialog segment in the air travel planning domain

…
AGENT 21: get a 71C
CLIENT 22: ok yeah and which stop do i get off at
AGENT 23: get off at /um/ forbes downtown forbes_and_stanwick down by the mcdonald's
CLIENT 24: yeah ok
AGENT 25: you will cross
CLIENT 26: yeah
AGENT 27: stanwick_street and you will see on your right after you pass the subway station /feed/ on the right on the left side will be the state office building and on the other side of the street will be the hilton hotel so just so you have your bearings so you know where you're goin
CLIENT 28: yeah #begin_feed# ok #end_feed#
AGENT 29: just walk straight up liberty_avenue and it should be right there on your left
CLIENT 30: ok
…

Figure 3.39: An example of a non-task dialog segment in the bus schedule inquiry domain

Among five task-oriented domains that can be represented with the form-based dialog structure, real-world applications (air travel planning, bus schedule enquiry and meeting) have more non-task segments than simulated tasks (map reading and UAV flight simulation) because the real-world applications are embedded in the world where a domain boundary is not clearly defined while the simulated tasks have a more limited scope that is well-defined. Since a meeting usually contains a lot of discussions, a conversation in the meeting domain contains more non-task segments than other types of task-oriented domains. Examples of non-task segments in the meeting domain are given in Section 3.6.

To verify that the form-based dialog structure can account for most parts of task-oriented dialogs on average, a dialog structure experiment (see Section 4.1) which determines the percentage of dialog content that can be accounted for by the proposed form-based dialog structure was conducted.

3.8.4 Potential difficulties in other task-oriented domains

The form-based dialog structure representation could be applied to other task-oriented domains besides the ones discussed in this chapter as well if the characteristics of dialogs in those domains match all of the assumptions made by the form-based representation. The form-based dialog structure framework assumes that 1) a goal of a dialog is achieved through the execution of domain actions, and 2) dialog participants have to communicate the information required to perform these actions through dialog. However, the form-based dialog structure representation may not be able to describe all the phenomena occurring in those task-oriented domains because it only captures the observable structure of a dialog using a simple model.

In addition to the difficulties discussed in the previous sections, a negotiation sub-dialog is another complicated case that is difficult to handle using the form-based representation. In the domains analyzed in this chapter, a negotiation only occurs at a concept level where dialog participants discuss alternative values for a particular concept rather than at a form level where the participants discuss alternative ways of performing one particular action. For example, in the air travel planning domain, a client may ask for an alternative flight, such as an earlier flight, from a set of flights that match the specified criteria as shown in utterance 10 -15 in Figure 3.12. This sub-dialog only discusses alternative values of a particular **FlightInfo** slot in a flight reservation form not alternative flight reservation forms. In all five domains analyzed (air travel planning, bus schedule inquiry, map reading, UAV flight simulation, and meeting), dialog participants help each other fill one particular form rather than proposing alternative forms since each participant has different pieces of information that need to be put together in order to perform a domain action or there is only one person who is responsible for making a decision on how to perform the action.

Nevertheless, in some domains, dialog participants may have similar pieces of information and help each other find the best way to perform an action. For example, in a scheduling domain, dialog participants have to find a meeting date and time that fit with all of the participants’ schedules. They may discuss several alternatives; each of them corresponds to a form rather than one specific slot. Even though the participants will choose only one alternative at the end, during a negotiation sub-dialog we may need multiple instances of the same form to represent all the options that are discussed. The proposed form-based dialog structure representation which assumes that there is only one instance of a form in each sub-task, while sufficient for modeling a negotiation at a concept level, is not adequate for modeling a negotiation at a form level where multiple form instances are required in each sub-task to represent alternative ways to perform the action that is associated with the sub-task.

We could extend the form-based representation to allow multiple instances of a form in each sub-task. However, the implication of this extension on domain-specific information acquisition and dialog processing should also be considered. For domain-specific information acquisition as in the case of this thesis, since there are multiple sets of concepts discussed in one sub-task (one set for each form instance), it is more difficult to identify a set of slots that is associated with one form especially if the concepts that belong to the same form instance are not discussed together in one segment. For dialog processing, an understanding module has to be able to identity the instance of a form that a dialog participant refers to in each utterance. This problem is more complicated than recognizing the participant’ focus of attention when a sub-task is interrupted since in a negotiation sub-dialog the focus of attention shifts within the same sub-task rather than between sub-tasks. For a more subtle shift in the focus of attention, usually there is less evidence that indicates the shift. Rosé et al. (1995) proposed an extension to a plan-based model that is able to capture multi-threads of negotiations in the scheduling domain. Nevertheless, dialog processing is not the main focus of this thesis.

3.9 Conclusion

In this chapter, I proposed a form-based dialog structure representation as a suitable representation for representing the domain-specific information required to build a task-oriented dialog system. The form-based dialog structure representation models the *tasks* that a dialog system has to support, a set of *sub-tasks* (a decomposition of a task) which corresponds to the steps that needs to be taken in order to successfully accomplish the task, and *concepts* which are the items of information (or domain keywords) that dialog participants have to communicate in order to achieve a task or a sub-task. In this framework, the domain-specific information in each dialog (i.e. tasks, sub-tasks, and concepts) is represented by forms and their slots while each utterance in the dialog is considered as an operator that operates on the forms and their content.

This form-based dialog structure representation is conjectured to have all of the required properties, sufficiency, generality, and learnability, discussed at the beginning of this chapter. The form-based representation is *sufficient* for representing the domain-specific information required to build a task-oriented dialog system since it is already used in the dialog systems that are implemented based on the form-based dialog system architecture, for example, the Philips train timetable information system (Aust et al., 1995) and the CMU Communicator system (Rudnicky et al., 1999). In this thesis, the notion of *form* in the form-based dialog structure representation is *generalized* beyond a database query form, so that it can be used to represent other types of task-oriented domains besides the information-accessing domains. The form-based dialog structure representation focuses only on concrete information that can be observed directly from in-domain conversations and models this information with a rather simple representation. These characteristics of the form-based representation (observable and simple) make it possible to be *learned* through an unsupervised learning approach.

Dialog structure analysis of six different types of task-oriented conversations described in this chapter demonstrates that the proposed form-based representation fulfills two of the required properties: *sufficiency* and *generality*. Interesting findings from the analysis are summarized below. The *learnability* of the form-based representation can be verified by the accuracy of its components obtained from the proposed machine learning algorithms described in Chapter 5 and Chapter 6.

By focusing only on observable aspects of a task-oriented dialog, the form-based dialog structure representation requires that all of the domain-specific information necessary for supporting a task is communicated clearly in a conversation. This occurs when a dialog has the following characteristics: 1) the dialog goal is achieved through the execution of domain actions, and 2) dialog participants have to communicate the information required to perform these actions through dialog. From dialog structure analysis of six task-oriented domains discussed in Section 3.2 - Section 3.7, I found that dialogs from five domains have the required characteristics and can be modeled by the form-based representation. These domains are air travel planning (information-accessing), bus schedule inquiry (information-accessing), map reading (problem-solving), UAV flight simulation (command-and-control), and meeting. However, the proposed dialog structure is not suitable for representing dialogs in the tutoring domain since their characteristics do not match the assumptions made by the form-based dialog structure representation. Furthermore, the simplicity of the form-based representation makes it not suitable for representing a complex dialog that has a dynamic structure as in the case of a tutor dialog.

Since human-human conversations are rich in nature and the form-based dialog structure representation only captures the observable structure of a dialog using a simple model, the form-based representation is not able to describe all of the phenomena occurring in those five task-oriented domains that it can be applied. The problematic cases include implicit concept values, fragmented sub-tasks, and non-task dialog segments. Nevertheless, the form-based dialog structure is sufficient enough to model important phenomena that occur regularly in dissimilar types of task-oriented dialogs, and thus has both *sufficiency* and *generality* properties. Even though a dialog system built from imperfect domain-specific information will have limited functionalities when compared to a human participant, it still be able to carry out the required tasks as demonstrated by the success of the systems that were implemented based on the form-based architecture. Dialog coverage, which measures the percentage of dialog content that can be accounted for by the proposed dialog structure and is reported in Section 4.1, is additional evidence that supports the *sufficiency* of the form-based dialog structure representation.

The form-based dialog structure representation also has another desirable property, a straightforward connection between its components and the components of a dialog system that employs the representation. Since we take a different approach using an existing dialog system framework to describe the structure of a task-oriented conversation rather than adapting an existing dialog structure theory to the existing dialog system architecture, configuring a dialog system from the acquired representation is less complicated as a mapping between form-based dialog structure components and the components of a form-based dialog system is straightforward. For instance, the learned domain-specific information can be used as a task specification in the RavenClaw framework (Bohus and Rudnicky, 2003).

The *learnability* of the form-based dialog structure representation can be verified by the accuracy of the tasks, sub-tasks, and concepts obtained from the learning algorithms proposed in Chapter 5 and Chapter 6. Implicit concept values and fragmented sub-tasks discussed in Section 3.8 may complicate a dialog structure acquisition process. However, if these complex cases only occur occasionally, the acquisition process should not be affected since the learning approaches which will be used to infer the components of the form-based dialog structure are based on the generalization of recurring patterns. If there are enough instances that clearly represent a particular dialog structure component, we should be able to learn the underlying concept or sub-task. Experiment results confirm this hypothesis; the learning accuracies of frequent concepts and sub-tasks are usually higher than the learning accuracies of infrequent components. The overall accuracy is acceptable for all learning problems: concept identification (see Chapter 5), and sub-task boundary detection and sub-task clustering (see Section 6.1 and 6.2 respectively) which are two main steps in form identification.

Learnability is also suggested if the structure of task-oriented dialogs can be marked up reliably with the proposed scheme. Section 4.2 describes a human annotation experiment that was carried out to evaluate the reliability of the proposed form-based dialog structure representation along two aspects: reproducibility and accuracy. Reproducibility measures the level of agreement among non-expert coders and verifies that the proposed dialog structure can be understood and applied by the annotators other than a coding scheme developer while accuracy ensures that the agreement between the coders conforms to the expert's judgment. Experiment results show that the form-based dialog structure annotation scheme can be understood and applied reliably by non-expert users producing high level of reproducibility and acceptability. High annotation scheme reliability suggests that the annotation scheme is concrete and unambiguous which imply learnability

Chapter 4

Form-based Dialog Structure Representation Evaluation

The previous chapter showed that the form-based dialog structure representation can be used to model the structure of dialogs in a variety of task-oriented domains. Nevertheless, to be useful for automatic acquisition of domain-specific information which is the goal of this thesis, the representation should also provide good coverage of dialogs, and, more importantly, it should be clear and unambiguous in its application. This chapter describes two annotation experiments that were carried out to validate these properties of the proposed form-based dialog structure representation.

To verify that the form-based dialog structure representation can account for most parts of task-oriented dialogs on average, a dialog structure experiment which determines *dialog coverage*, the percentage of dialog content that can be accounted for by a given dialog representation, was conducted. This experiment, which will be described in Section 4.1, also identifies those pieces of human-human conversation that the proposed dialog structure fails to account for. This evaluation procedure verifies two required properties of a domain-specific information representation (*sufficiency* and *generality*) discussed in Chapter 3. If the proposed representation achieves high dialog coverage in various types of task-oriented domains, we can say that the representation fulfills both the sufficiency and generality requirements.

In addition to being able to model important phenomena in dissimilar types of task-oriented dialogs, a domain-specific information representation has to be concrete and unambiguous, so that it can be identified reliably from in-domain dialogs. *Learnability*, which is another desired property, is implied if the structure of task-oriented dialogs can be marked up reliably by annotators other than coding scheme developers using the proposed dialog structure annotation scheme. A dialog structure annotation experiment described in Section 4.2 was conducted in order to evaluate the form-based dialog structure representation along two aspects of annotation scheme reliability: *reproducibility*, which requires different annotators to produce similar annotations, and *accuracy*, which requires an annotator to produce a similar annotation as a known standard (e.g. a coding scheme expert's annotation).

4.1 Dialog coverage

Dialog coverage is defined as the percentage of dialog content that can be accounted for by a specific dialog structure. In the form-based dialog structure framework, domain-specific information in each dialog is represented by forms. Each utterance in the dialog is regarded as an operator that operates on the forms and their content. Therefore, an utterance that can be classified as one of the form operators discussed in Section 0 is an utterance that can be described by the form-based dialog structure representation. Dialog coverage for the form-base representation is computed by the following equation.

$$dialog\ coverage = \frac{number\ of\ utterances\ \ that\ can\ be\ described\ \ by\ the\ structure}{total\ number\ of\ utterances} * 100 \qquad (4.1)$$

To measure the dialog coverage of the proposed form-based dialog structure representation, four task-oriented domains: air travel planning, bus schedule enquiry, map reading and UAV flight simulation were analyzed on an utterance-by-utterance basis. Table 4.1 shows the number of dialogs that were available during the time of the evaluation as well as those used in the analysis.

Domain	Available	Analyzed	
	Number of dialogs	**Number of dialogs**	**Number of utterances**
Air travel planning	43	4	273
Bus schedule enquiry	12	5	90
Map reading	128	4	498
UAV flight simulation	2	1	224

Table 4.1: The amount of data used in the analysis

On average, 93% of the utterances that were analyzed could be accounted for by the form-based dialog structure representation. Utterances that could not be described by the proposed framework are classified into four categories: out of domain (OOD), out of scope (OOS), indirect and task management (TM).

1. *Out of domain (OOD)* utterances contain information that does not relate to the domain (and thus would not need to be accounted for).

2. *Out of scope (OOS)* utterances contain information that while related to the domain, falls outside the scope of the defined conversation goal, or does not contribute directly toward an execution of any domain action. For example, in the bus schedule inquiry domain, a dialog goal is to obtain information about a bus schedule; therefore, the utterances that contain an additional discussion about how to get to a client's actual destination from a bus stop are out of scope. More discussion about this type of utterance can be found in Section 3.8.3.

3. *Indirect utterances* are utterances that require additional domain and world knowledge that are not presented in a dialog in order to interpret and act upon. For example, in the bus schedule enquiry domain, instead of specifying the actual time that he would like to arrive at the airport, a client may provide the departure time of his flight as an implicit arrival time of the desired bus. Additional knowledge representation and interpretation process are required in order to represent this type of utterance with the form-based dialog structure representation. However, this type of utterance does not include the utterances that contain indirect expressions which can be resolved using contextual information from within a dialog such pronouns and indirect temporal expressions. Indirect expressions are discussed in more detail in Section 3.8.1.

4. *Task management (TM)* utterances are utterances that manage the overall state of a dialog (e.g. suspending or resuming a task) rather than manipulate a particular form. This type of utterance occurs when a sub-task is interrupted by another sub-task. "Can you hold one second please" is one example of TM utterances. While a task management utterance contain information that is necessary for run time dialog processing such as determining a speaker's focus of attention, it does not contain information that is necessary for performing any domain action. The form-based dialog structure representation focuses on modeling domain-specific information required in order to execute domain actions that contribute directly toward a dialog goal rather than the information that may be required to process the dialog at run time as discussed at the end of Section 3.8.2.

Domain	Rejected utterances (%)				
	OOD	OOS	Indirect	TM	Total
Air travel planning	4.4	4.4	6.7	0.0	15.6
Bus schedule enquiry	1.8	4.4	0.4	2.6	9.2
Map reading	0.0	0.0	2.2	0.0	2.2
UAV flight simulation	1.0	0.0	1.0	4.0	5.9

Table 4.2: The percentage of rejected utterances of each type

The percentage of rejected utterances in Table 4.2 reveals that the characteristics of a task affect both the type and the number of rejected utterances. The air travel planning and the bus schedule enquiry which are real-world applications have higher rejected utterance rates than the map reading and the UAV flight simulation which are simulated tasks. The main difference comes from the scope of the task as reflected in the number of OOS. The simulated task has a limited scope that is well-defined while the real-world application is embedded in the world and appears to have an indefinite domain boundary. Since a dialog system that is applied to the real-world task will have limited functionalities, we would expect it to elicit more OOS utterances.

Nevertheless, previous studies on the differences between human-human conversations and human-machine conversations (Dahlbäck et al., 1993; Hauptmann and Rudnicky, 1988; Jönsson and Dahlbäck, 1988) show that the language that a human uses to communicate with a computer is more constrained than the one he/she uses to communicate with a human participant. Since humans can adjust their language to accommodate the machine incomplete communication capability, we would expect to see fewer rejected utterances in human-machine conversations as the humans constrain themselves within the scope of a dialog system application and use simple and more direct utterances to communicate with the system.

Summary

The proposed form-based dialog structure representation has high *dialog coverage* in all four task-oriented domains (air travel planning, bus schedule enquiry, map reading, and UAV flight simulation) used in the evaluation. This result verifies that the form-based representation has both the *sufficiency* and *generality* properties. Dialog coverage is lower in real-world applications (bus schedule enquiry and air travel planning) than in simulated tasks (map reading and UAV flight simulation) due to the greater difference between the broader scope of the real-word application and the limited dialog system functionalities.

4.2 Annotation scheme reliability

The evaluation result in the previous section shows that that the proposed form-based dialog structure representation can account for large portions of dialogs in task-oriented domains. However the evaluation was carried on by an annotation scheme developer. It is also important to show that the proposed dialog structure can be understood and applied reliably by annotators other than the developer. In this section, I will discuss the evaluation of the form-based dialog structure representation along two aspects of reliability: *reproducibility* and *accuracy*. Reproducibility, which measures inter-coder variance, requires different annotators to produce similar annotations while accuracy requires an annotator to produce a similar annotation as a known standard (e.g. a coding scheme expert's annotation). The evaluation was done through a dialog structure annotation experiment.

First, a set of pilot annotation experiments was conducted in order to verify the experimental procedure and the annotation guidelines that would be used in the annotation experiment. Pilot experiments are described in Section 4.2.1. The form-based dialog structure annotation scheme poses two challenges in comparing two dialog structure annotations. Firstly, different tagsets may be used to annotate dialog structures in the same domain. Secondly, structural annotations are quite difficult to compare. These two challenges are discussed in Section 4.2.2 and Section 4.2.3 respectively. The experimental procedure of the annotation experiment is discussed in Section 4.2.4 and the results are presented in Section 4.2.5. Finally, Section 4.2.6 summarizes all the findings.

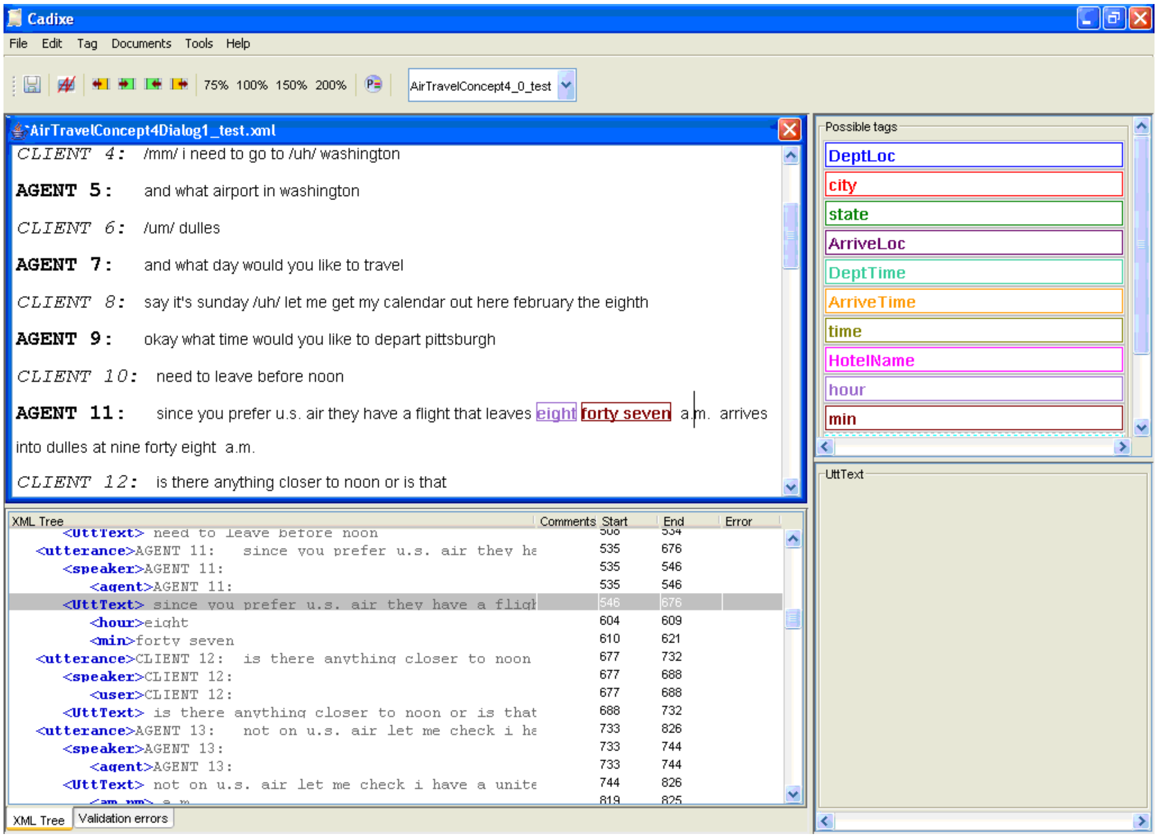
4.2.1 Pilot annotation experiments

A set of pilot annotation experiments was conducted to verify an experimental procedure especially annotation guidelines. Five subjects were participated in the pilot experiments. Since we would like to verify that the proposed form-based dialog structure representation can be understood and applied reliably by annotators other than a coding scheme developer, eligible subjects must not used the form-based annotation scheme previously. I focus particularly on a group of people who are likely to use the form-based dialog structure representation in the future. This target group includes people who are involved in dialog system development, linguistic and language technologies students.

In the experiments, the subjects were asked to annotate the structures of the given dialogs using the form-based dialog structure annotation scheme. Coding scheme reliability is determined from the similarity of the annotations produced by different coders. The subjects were given annotation guidelines which contain a form-based dialog structure definition and examples of dialog structure annotations from the task-oriented domains that were not used in the experiments. Since the form-based annotation scheme only specifies the definition of each dialog structure component (task, sub-task, and concept), but does not prescribe a specific tagset to be used in each domain, each annotator has to also develop his/her own tagset. In the annotation experiment, each subject had to first design a dialog structure representation (a tagset or a markup scheme) for a given domain according to the definitions of task, sub-task, and concept provided in the guidelines by analyzing a given set of in-domain dialogs. The markup scheme includes the tags for tasks, sub-tasks and concepts for the given domain. The subject was then asked to annotate a set of dialogs according to the tagset that he/she had designed. This set of dialogs is the same set as the one the subject saw when designing the dialog structure representation. This procedure is different from a typical annotation task in which the tagset is pre-specified and is given to the annotators. To control the number of annotated instances for each dialog structure component and to keep the annotation experiment simple, the experiment was divided into two parts: concept annotation, and task/sub-task annotation. There was no time limit on any part of the experiment.

The annotation was done using CADIXE[1], an XML-based annotation tool. CADIXE provides easy and interactive environment for text document annotation. An annotator is not required to have any experience with XML annotation as an annotated document is displayed in text format (in different colors and styles depending on the markups) in the annotation panel while its corresponding XML representation is displayed in the XML panel. CADIXE user interface is illustrated in Figure 4.1. Since each annotation tag has it own display style in the annotation panel, it is easy to distinguish between different types of dialog structure components. For example, **Hour** and **Minute** are displayed with different colors as shown in Figure 4.1. CADIXE version 2.0a6 was used in all of the annotation experiments.

[1] More detail about CADIXE can be found at http://www-leibniz.imag.fr/SICLAD/Caderige/Cadixe/



```
CLIENT 4:   /mm/ i need to go to /uh/ washington

AGENT 5:    and what airport in washington

CLIENT 6:   /um/ dulles

AGENT 7:    and what day would you like to travel

CLIENT 8:   say it's sunday /uh/ let me get my calendar out here february the eighth

AGENT 9:    okay what time would you like to depart pittsburgh

CLIENT 10:   need to leave before noon

AGENT 11:    since you prefer u.s. air they have a flight that leaves eight forty seven  a.m.  arrives
into dulles at nine forty eight  a.m.

CLIENT 12:   is there anything closer to noon or is that
```

```
XML Tree                                                 Comments Start   End    Error
      <UttText> need to leave before noon                         508     534
   <utterance>AGENT 11:   since you prefer u.s. air they he       535     676
      <speaker>AGENT 11:                                          535     546
         <agent>AGENT 11:                                         535     546
      <UttText> since you prefer u.s. air they have a fligh       546     676
         <hour>eight                                              604     609
         <min>forty seven                                         610     621
   <utterance>CLIENT 12:  is there anything closer to noon        677     732
      <speaker>CLIENT 12:                                         677     688
         <user>CLIENT 12:                                         677     688
      <UttText> is there anything closer to noon or is that       688     732
   <utterance>AGENT 13:   not on u.s. air let me check i he       733     826
      <speaker>AGENT 13:                                          733     744
         <agent>AGENT 13:                                         733     744
      <UttText> not on u.s. air let me check i have a unite       744     826
         <am_pm> a.m.                                             819     825
```

Figure 4.1: The interface of CADIXE XML annotation editor

After each pilot experiment, potential problems and ambiguities in both the experimental procedure and the annotation guidelines were identified by examining the tagset and the annotated dialogs that each subject produced. The subjects were also asked to explain the difficulty or ambiguity that they may have encountered during the experiment. The annotation guidelines were then successively modified to address those problems. The modifications include:

- More precise definitions of dialog structure components
- Clarifications for ambiguous components and more contrasting examples

The following pieces of information were also added:

- A formal definition of an action as it plays an important role in the decomposition of tasks and sub-tasks
- A diagram that illustrates the decomposition of a dialog into tasks and sub-tasks together with their associated actions and forms in each domain
- A table that summarizes dialog structure components in each domain
- More examples of annotated dialogs

In terms of the experimental procedure, more detail instructions were given especially on how the annotated dialog structure would be used (i.e. to build a dialog system based on the information in the dialog structure). An annotation exercise was also added to assess subjects' understandability on the form-based dialog structure annotation scheme. After the subjects studied annotation guidelines, they were asked to annotate the structures of dialogs in the same domains as the ones discussed in the guidelines. In the exercise, the tagsets were provided for both domains (the tagsets are described in the guidelines); the subjects did not have to design a new tagset. Only a subject who sufficiently understood the form-based dialog structure annotation scheme would be entered into an annotation experiment.

The pilot experiments revealed that the subjects could come up with dissimilar dialog structure designs (dissimilar markup schemes) even though they were working on the same set of dialogs. Table 4.3 shows examples of the differences found in the air travel planning domain. These differences include: additional concept types and variations in structured concept design. Annotation differences and the possible causes are discussed in more detail in Section 4.2.5.1 with the data from the main experiment.

Subject 1	Subject 2
<NoOfStop>	-
<DestinationCity>	<DestinationLocation><City>
<Date>	<DepartureDate> and <ArrivalDate>

Table 4.3: Differences in the tagsets designed by two different subjects

Variations in the tagsets are acceptable as long as the tagsets and the annotations conform to the dialog structure definition given in the guidelines. Moreover equivalent but not identical dialog structures can generate dialog systems with the same functionality. This is analogous to different computer programs that generate the same output. The variations in the tagsets, however, create a challenge in evaluating inter-annotator agreement as the conventional inter-annotator agreement metric assumes that an identical tagset is used by all annotators. To verify that an annotator can generate a reasonable domain-specific tagset for a given task-oriented domain rather than one specific tagset that matches a known answer, an evaluation methodology that can accommodate differences in annotation tagsets is required. To solve this problem, I propose a novel evaluation procedure which will be referred to as *cross-annotator correction*. The detail of this procedure is discussed in the next section.

4.2.2 Cross-annotator correction

I propose *cross-annotator correction* as a suitable technique for assessing inter-annotator agreement when the annotators create different markup schemes to annotate the same document. Instead of comparing two annotated documents produced by two different annotators mechanically, each annotated document is first critiqued and corrected by another annotator (a corrector). The corrector makes judgment based on the tagset created by the original annotator of that document and is only allowed to correct the annotation when it does not conform to the form-based dialog structure annotation guidelines, but not when it differs from his/her own annotation. The original annotation and its corrected version are then compared. Both annotated documents in cross-annotator comparison are based on the same (original) tagset. Figure 4.2 illustrates the cross-annotator correction process. The cross-annotator comparison allows the original annotation to be evaluated against not only a single annotation but any annotations that conform to the guidelines. Thus, the idea is to compare annotators' knowledge of the annotation scheme and not individual instances of annotation.

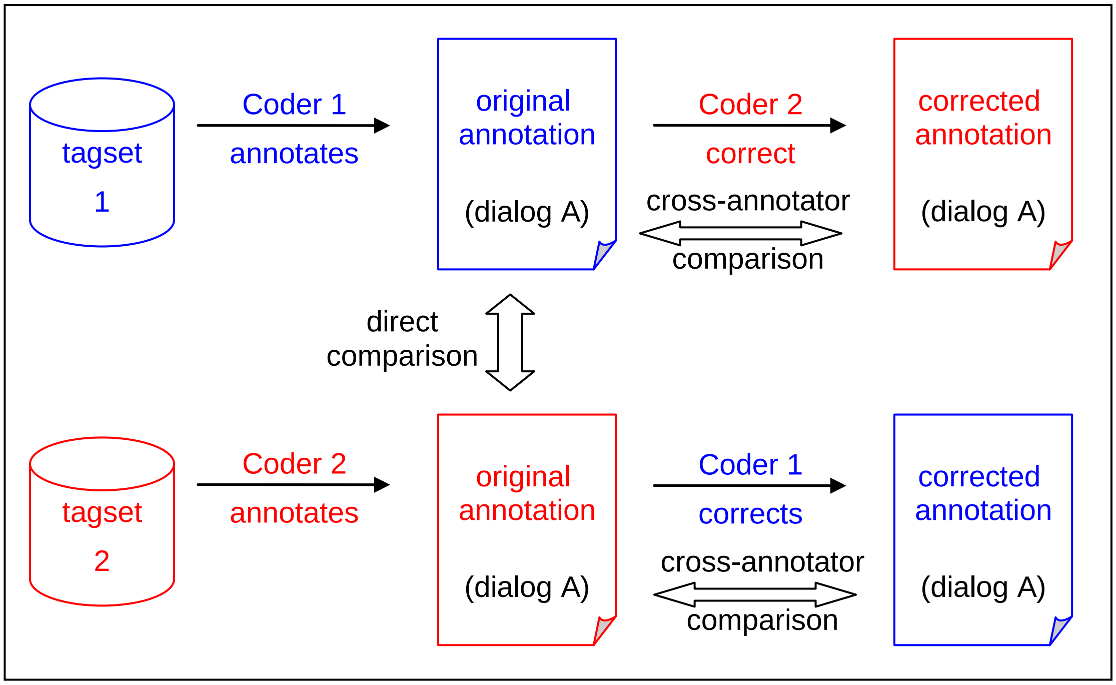


Figure 4.2: A cross-annotator correction process

A procedure for evaluating the reliability of the form-based dialog structure representation using the cross-annotator correction process is as follows. Each annotator is first asked to develop his/her own annotation tagset and use it to annotate the dialog structures of the given dialogs. This part is similar to the dialog structure annotation experiment described in Section 4.2.1. The part of the procedure creates the original annotations shown in Figure 4.2. In the second part of the procedure, the annotator (the corrector) is asked to critique and correct the annotation done by another annotator that is given the same instruction and annotation guidelines and is working on the same set of dialogs. The corrector is instructed to correct the annotation only when it does not conform to the form-based dialog structure annotation guidelines not when it differs from his/her own annotation.

The correction process consists of two steps: annotation critique and annotation correction. In the first step, the corrector is asked to rate each dialog structure component in the original annotation as correct or incorrect. I am also interested in the corrector's level of confidence when making each decision because the corrector may agree or disagree with the original annotation when he/she is not certain with the decision. Therefore, instead of judging each annotation tag as only correct or incorrect the corrector is also asked to provide the level of confidence in his/her critique by rating each tag using one of the four possible correctness values: *correct*, *maybe correct*, *maybe incorrect* and *incorrect*.

To ensure that the corrector marks an original tag as incorrect or maybe incorrect only when it does not conform to the guidelines, the corrector has to also specify the error category of that incorrect tag. In the main dialog structure annotation experiment described in Section 4.2.4, a list of common types of errors found in concept annotation, and a list of common types of errors found in task and sub-task annotation were given to the subjects to provide an idea of what kinds of annotation mistakes they might encounter. These lists were created by a coding scheme expert from the analysis of the annotation errors found in the pilot experiments. Nevertheless, the subjects could also correct additional types of errors when they discovered ones. The lists of common error types that were given to the subjects are shown in Figure 4.3 and Figure 4.4. The name in the parentheses is an error code which can be used interchangeably with an error type. Examples of task and sub-task annotation errors were taken from the bus schedule enquiry domain. The subjects could use the criteria in the following diagrams as guidelines to determine an error type of each incorrect tag. A diagram in Figure 4.5 demonstrates how to identify the type of error in concept annotation while a diagram in Figure 4.6 demonstrates how to identify the type of error in task and sub-task annotation.

1. **Missing concepts**

   The annotator did not annotate **crucial information** that is required in order to perform an action. There are two cases that a concept **C** could be missing from the annotation.

   1.1. Missing concept type (**MissingType**): the annotator did not identify the concept **C.** That is **C** is not in the list of concepts.

   1.2. Missing concept instance (**MissingInstance**): the annotator did not annotate an instance of **C** when it is presented in the dialog even though he/she has **C** in his/her concept list.

2. **Extraneous concept types (ExtraType)**

   The annotator identified a concept that is not relevant to a task or is not necessary for performing an action.

3. **Incorrect use of concept labels (IncorrectLabel)**

   Use **C** to annotate a word or a group of words that does not belong to a concept **C**. This includes 1) annotate a word that belong to another concept as **C** and 2) annotate a word that doesn't belong to any concept (a non-concept word) as **C**.

4. **Incorrect concept boundary (IncorrectBoundary)**

   Only a part of a concept is annotated, for example:

   (incorrect) <e-mail>tom</e-mail>@cmu.edu

   (correct) <e-mail>tom@cmu.edu</e-mail>

5. **Inappropriate structured concept granularity (IncorrectGranularity)**

Use a too fine-grained or a too coarse-grained structured concept. For instance, the annotator annotated a structured concept without annotating its components when the individual component can be used separately, e.g. if a zip code is used separately for calculating shipping cost then,

```
(incorrect)   <address> 5000 Forbes Ave.
                  Pittsburgh PA 15213
              </address>
(correct)     <address> 5000 Forbes Ave.
                  Pittsburgh PA <zipcode>15213</zipcode>
              </address>
```

6. **No distinction between similar concepts that have different functionalities (NoDistinction)**

   For example, in an e-mail application, it is **incorrect** to annotate all e-mail addresses with the same concept type **Address**. The **correct** annotation is to distinguish between **SenderAddress** and **RecipientAddress**

Figure 4.3: A list of common types of errors found in concept annotation

1. **Missing tasks or sub-tasks**

   The annotator did not annotate a subset of a dialog that corresponds to a task or a sub-task. There are two cases that a task or a sub-task could be missing.

   1.1. Missing a task/subtask type (**MissingType**)

   1.2. Missing a task/sub-task instance (**MissingInstance**)

2. **Extraneous tasks or sub-tasks (ExtraComponent)**

   An extraneous task is a sub-set of a dialog that does not have a specific goal. An extraneous sub-task is a sub-set of a dialog that does not correspond to any well-defined action including too fine-grained decomposition of a task where each sub-task corresponds to acquiring each piece of information. For example, a get_bus_number sub-task and a get_departure_location sub-task, which are decomposed from a **query_departure_time** task, are **extraneous sub-tasks**. Each of them does not correspond to any action by itself, but rather acquires one piece of information for an action ***retrieve_depart_time_fromDB***.

3. **Incorrect use of task or sub-task labels (IncorrectLabel)**

   Use **T** to annotate a subset of a dialog that does not correspond to a task **T** or use **S** to annotate a subset of a dialog that does not correspond to a sub-task **S**.

4. **Incorrect task or sub-task boundary (IncorrectBoundary)**

   For example, including a part of a dialog that belongs to the adjacent task in the scope of the considered task.

5. **Inappropriate task or sub-task granularity (IncorrectGranularity)**

   Examples of this type of error are:

   5.1. Annotating multiple tasks as one task

   The following case is considered an annotation **errors**; the annotator annotated a dialog that discusses two independent bus schedule queries as one task. The **correct** annotation must separate the dialog into two tasks; one for each query.

   5.2. Merging adjacent sub-tasks together

   A sub-task that contains more than one action **does not conform** to the guideline.

6. **No distinction between different types of tasks or different types of sub-tasks (NoDistinction)**

   Subsets of a dialog that have dissimilar types of goals should be annotated with different task labels. Similarly, subsets of a dialog that correspond to dissimilar types of actions should be annotated with different sub-task labels

7. **Inappropriate task or sub-task decomposition (IncorrectDecomposition)**

   This includes identifying a sub-task as a task, a task as a sub-task, a second-level sub-task as a first-level sub-task and so on.

   Tasks are independent from each other while the sub-tasks of the same task have some dependencies among them. Similarly, there are more dependencies between sub-tasks in the same decomposition level than between sub-tasks in different levels that do not subsume one another. The annotation that marks a subset of a dialog that has some dependencies with other parts of the dialog as a task **does not conform** to the guideline.

Figure 4.4: A list of common types of errors found in task and sub-task annotation

```
for each original tag T which covers a word or a group of words Ws
    if a word or a group of words Ws should not be annotated as a concept then
        /*an extraneous label was used (e.g. a non-concept word was annotated)*/
        if every instance of the concept T should not be annotated then
            Error Type = ExtraType
            correction = 1. Remove T from the concept list (change the annotation scheme)
                         2. Remove T from the annotation
        else
             /*just some instances of T are incorrect*/
            Error Type = IncorrectLabel
            correction = 1. Remove T from the annotation
        endif
    else
        if T is not the right tag for this concept then
            if there is a tag R in the original concept list that should be used instead then
                Error Type = IncorrectLabel
                correction = 1. Change T to R
            else
                /*an appropriate concept type is missing*/
                if the missing concept type M is required in order to distinguish between similar concepts
                  that have different functionalities then
                    Error Type = NoDistinction
                else
                    Error Type = MissingType
                endif
                correction = 1. Add the missing concept type M into the concept list (change the annotation
                                scheme)
                             2. Change T to M
            endif
            correction = (optional) adjust the boundaries of the new tag if necessary
        else
            /* T is the right tag for this concept */
            if T is a structured concept and its granularity is inappropriate then
                Error Type = IncorrectGranularity
                correction = 1. Modify the structured concept components in the annotation scheme which
                                may include adding a missing component
                             2. Modify the annotation (the error type of the added component should be
                                MissingType or MissingInstance)
            else if the concept boundary is incorrect then
                Error Type = IncorrectBoundary
                correction = 1. Adjust the boundary
            else if T contains another type of error then
                Error Type = Other (please also describe the new error type in the Description attribute)
            else
                This tag is correct
            endif
        endif
    endif
end for each
```

Figure 4.5: Criteria for classifying an error type in concept annotation

```
for each original tag $T$ which covers a subset of a dialog $D$
  if $T$ was placed at an inappropriate position in the task structure hierarchy (e.g. $T$ was incorrectly
    identified as a task instead of a sub-task) then
      Error Type = IncorrectDecomposition
      correction = 1. Modify the task structure hierarchy (change annotation the scheme)
                   2. Modify the annotation; let $P_C$ be the current parent tag of $T$ according to the
                      original task structure hierarchy and $P_N$ be the new parent tag of $T$ according to
                      the modified task structure hierarchy
                      2.1 To move $T$ down the hierarchy (e.g. change from task to sub-task)
                          ○ Extend the boundaries of $P_N$ to include $T$ ($P_N$ should be marked as
                            IncorrectBoundary)
                          or
                          ○ Add $P_N$ if it is missing (an error type of $P_N$ should be MissingInstance or
                            MissingType)
                      2.2 To move $T$ up the hierarchy (e.g. change from sub-task to task)
                          ○ Adjust the boundaries of $P_C$ to exclude $T$ ($P_C$ should be marked as
                            IncorrectBoundary)
                          or
                          ○ Remove $P_C$ ($P_C$ should be marked as ExtraComponent)
  else
    Identify the number of goals $N_G$ in $D$ if $T$ is a task or
    Identify the number of actions $N_A$ in $D$ if $T$ is a sub-task (do not including the actions that
        correspond to the embedded sub-tasks of $T$ in $N_A$)
    if $N_G$ or $N_A = 0$ then
        /* D contains no specific goal or action */
        Error Type = ExtraComponent
        correction = 1. Remove $T$ from the annotation
                     2. (optional) adjust the boundaries of adjacent tags if necessary (those
                        adjacent tags should be marked as IncorrectBoundary)
    else if $N_G$ or $N_A > 1$ then
        /* merged multiple tasks or sub-tasks together*/
        Error Type = IncorrectGranularity
        correction = 1. Remove tag $T$ from the annotation
                     2. Annotate all the merged components separately (the new components
                        should be marked as MissingType or MissingInstance)
```

Figure 4.6: Criteria for classifying an error type in task and sub-task annotation

```
    else if $N_G$ or $N_A$ = 1 then
        if T is not the right tag for this task or sub-task then
            if there is a tag R in the original tagset that should be used instead then
                Error Type = IncorrectLabel
                correction = 1. Change T to R
            else
                /*an appropriate tag is missing*/
                if the missing tag M is required in order to distinguish between different types of
                  tasks or different types of sub-tasks then
                    Error Type = NoDistinction
                else
                    Error Type = MissingType
                endif
                correction = 1. Add the missing tag M into the task structure hierarchy (change the
                                annotation scheme)
                             2. Change T to M
            Endif
            correction = (optional) adjust the boundaries of the new tag if necessary
        else
            /* T is the right tag for this concept */
            if the boundary of T is incorrect then
                Error Type = IncorrectBoundary
                correction = 1. Adjust the boundary
            else if T contains another type of error then
                Error Type = Other (please also describe the new error type in the Description
                              attribute)
            else
                This tag is correct
            endif
        endif
    endif
  endif
end for each
```

Figure 4.6: Criteria for classifying an error type in task and sub-task annotation (cont.)

In the second step of the correction process, annotation correction, a corrector has to make the original annotation conforms to the form-based dialog structure annotation guidelines by correcting all of the tags that he/she marks as *incorrect*. For the tag that is marked as *maybe incorrect*, the corrector has an option to correct it if he/she is certain about the correction or leave it unchanged if he/she is not so sure. The corrector can also insert the tags that are missing from the original annotation. For each new tag, the corrector has to specify the type of error he/she intends to correct by adding this particular tag along with the degree of confidence (*confident* or *not so confident*) in the insertion. In the dialog structure annotation experiment, the subjects could use the suggested corrections provided in the error type classification diagrams in Figure 4.5 and Figure 4.6 as guidelines for correcting each type of error.

4.2.3 Annotation similarity

In cross-annotator comparison we need to compare two structural annotations. Furthermore, since there is no limitation on the number of levels in structured concept nor in sub-task decomposition, the number of annotation levels in form-based dialog structure annotation can be arbitrary.

The Kappa coefficient (*K*) is a commonly used metric for assessing inter-annotator agreement. However, *K* is used primarily for comparing categorical judgments such as the dialog act label for each utterance. In order to use *K* with structural annotation, some extension is required. For instance, a cascaded approach was used in (Carletta et al., 1997; Moser and Moore, 1997). In this approach, structural annotation is broken down into levels and only one annotation level is considered at a time. The highest annotation level in the structural annotation is evaluated first, then only the next level tags that are under agreed tags are considered. By separating the calculation in this way, it is quite difficult to tell directly the level of agreement on the overall structure and it is also not suitable for a structural annotation that has arbitrary number of levels as in the form-based dialog structure representation. Other approaches (Flammia and Zue, 1995; Marcu et al., 1999) compared entire structural annotations directly without separating them into levels. However, the *K* values computed by these approaches may be artificially high. Flammia and Zue (1995) can only provide a lower bound estimation of chance agreement, thus an upper bound on *K* while Marcu et al.'s (1999) approach has a large numbers of non-active spans (a hierarchical structure of n segments is mapped into a set of $\frac{n(n+1)}{2}$ overlapping spans) which may boost the observed agreement. Although the kappa statistic makes a correction for chance agreement, its calculation still needs to be based on a sensible choice of unit (Carletta, 1996).

One can view a problem of annotating tasks, sub-tasks and concepts in a given dialog as a bracketing and labeling problem where a labeled bracket (task, sub-task or concept label) has to be placed around a word or a group of words. From this perspective, structural annotation of a dialog is analogous to structural annotation of a sentence. Brants (2000) and Civit et al. (2003) proposed similar inter-annotator agreement measures for syntactic structure annotation based on bracketed precision and bracketed recall. Both bracketed precision and bracketed recall were first introduced in the

PARSEVAL workshop as quantitative metrics for evaluating parser outputs (Abney et al., 1991).

Based on the ideas of Brants and Civit et al., I define *acceptability* as the degree to which an original annotation is acceptable to a corrector by measuring the similarity between an original annotation and its corrected version. If the original annotation and its corrected annotation are very similar, the acceptability value will be high. On the contrary, if the original annotation and its corrected annotation are very different, the acceptability value will be low. Let *X* be the original annotation and *X'* be the corrected annotation:

$$acceptability(X, X') = avg(precision(X, X'), recall(X, X')) \qquad (4.2)$$

where

$$precision(X, X') = \frac{\text{number of similar brackets}}{\text{total number of brackets in } X} \qquad (4.3)$$

$$recall(X, X') = \frac{\text{number of similar brackets}}{\text{total number of brackets in } X'} \qquad (4.4)$$

Two annotated tags (or brackets) are considered similar if both their contents and labels are the same. For example, <Date>may eleventh</Date> is not similar to <DepartureDate>may eleventh</DepartureDate>, and <Fare>fifty dollars</Fare> is not similar to <Fare>fifty</Fare>. This matching criterion is regarded as *exact match*. Alternative matching criteria will be discussed later in this section. Acceptability is an average between bracketed precision and bracketed recall. Both numbers are averaged using a harmonic mean similar to the way traditional precision and recall are combined to provide an F-1 measure. The bracketed precision and recall allows arbitrary levels in structural annotations to be compared at the same time.

The changes that a corrector made to an original annotation during the cross-annotator correction process can be classified into two board categories: major change and minor change. A *major change* modifiers the original dialog structure design, for example, adding a concept type or changing the composition of a structured concept, while a *minor change* occurs at the instance level and affects only the annotation and not the design, for example, changing a concept boundary or adding a missing annotation (instance not type). For task and sub-task annotation, a task or sub-task that corresponds to more than one action or does not correspond to any well-defined action is considered a major error. Major and minor are referred to as the *severity level* of a change.

The corrector makes changes to the original annotation in order to correct annotation errors. Therefore, each change is associated with an incorrect tag in the original annotation. The severity level of each change can be classified according to the error type of the incorrect tag that the corrector intends to correct. The terms *change, correction*, and *error* can be used interchangeably in this context. Common error types in Figure 4.3 and Figure 4.4 are shown again in Table 4.4 and Table 4.5 respectively along with their severity levels.

No.	**Error type**	**Severity level**
1.1	Missing concept types (MissingType)	major
1.2	Missing concept instances (MissingInstance)	minor
2	Extraneous concept types (ExtraType)	major
3	Incorrect use of concept labels (IncorrectLabel)	minor
4	Incorrect concept boundary (IncorrectBoundary)	minor
5	Inappropriate concept granularity (IncorrectGranularity)	major
6	No distinction between similar concepts that have different functionalities (NoDistinction)	major

Table 4.4: Severity levels of common errors in concept annotation

No.	**Error type**	**Severity level**
1.1	Missing task or sub-task types (MissingType)	major
1.2	Missing task or sub-task instances (MissingInstance)	minor
2	Extraneous tasks or sub-tasks (ExtraComponent)	major
3	Incorrect use of task or sub-task labels (IncorrectLabel)	minor
4	Incorrect task or sub-task boundary (IncorrectBoundary)	minor
5	Inappropriate task or sub-task granularity (IncorrectGranularity)	major
6	No distinction between different types of tasks or different types of sub-tasks (NoDistinction)	major
7	Inappropriate task or sub-task decomposition (IncorrectDecomposition)	major

Table 4.5: Severity levels of common errors in task and sub-task annotation

A major error is more severe because it affects every instance of that type whereas a minor error affects only a specific instance. For example, MissingType is a major error because all of the instances of that type are not annotated while MissingInstance is a minor error because only one specific instance of a dialog structure component is omitted.

Different dialog structures, which caused by major changes, may lead to dissimilar dialog systems. Some changes can cause the resulting dialog systems to have different functionalities. For instance, additional sub-tasks add extra actions to a dialog system. However, some differences can be compensated by a more sophisticated dialog system component. For example, a dialog system that built from a dialog structure that does not have detailed concept annotation (e.g. only contains <DepartLoc> not <city> and <state>) requires a more complex understanding module. Nevertheless, this dialog system still has the same functionality as a dialog system that built from a dialog structure that has detailed concept annotation. As it is quite subjective to decide how each change will affect dialog system functionalities, we merely consider both types of changes together as major changes rather than making subjective distinction between them. All of the minor changes will not affect the functionality of the target dialog system because they only change an annotation at the instance level but do not change a dialog structure design.

In Equation (4.3) and (4.4), two annotated tags are considered similar if both of them are matched under a specified criterion. A common matching criterion is an exact match discussed earlier. By taking into account two additional pieces of information: degree of correctness and severity level, more flexible matching criteria for comparing two dialog structure annotations (an original annotation and its corrected annotation) can be defined. Six matching criteria with different degrees of rigorousness are described below. Each of them is defined based on how an annotation corrector judges the original tag in terms of its degree of correctness and type of error (for an incorrect one). The most rigorous criterion is listed first.

1. *confident match:* two annotation tags are considered matched if a corrector rates the original tag as correct (not include maybe correct)

2. *agreed match:* two annotation tags are considered matched if a corrector rates the original tag as correct or maybe correct

3. *labeled match:* two annotation tags are considered matched if a corrector does not make any correction. This includes the case when a corrector rates the

original tag as maybe incorrect but leaves it unchanged. Labeled match is equivalent to exact match.

4. *labeled match and minor change:* two annotation tags are considered matched if a corrector does not make any correction or makes only a minor change
5. *labeled match and unconfident change:* two annotation tags are considered matched if a corrector does not make any correction or makes a change with uncertainty (the original tag is marked as maybe incorrect or not so confident)
6. *all but confident major change:* two annotation tags are considered matched in all cases except the case that a corrector makes a major change with confidence (the original tag is marked as incorrect or confident)

Criterion 4, 5 and 6 allow some acceptable changes to be considered as matched.

4.2.4 Annotation experiment procedure

The reliability of the proposed form-based dialog structure representation was evaluated in two types of task-oriented domains, air travel planning (information-accessing task) and map reading (problem-solving task). The detail descriptions of both domains are described in Section 3.2 and Section 3.4 respectively. Eight participants were drawn from a target group, a group of people who are likely to use the form-based dialog structure representation in the future, which includes people are who involved in dialog system development, linguistic and language technologies students. This target group is the same target group as the one used in the pilot experiments. The subjects were drawn primarily from the Carnegie Mellon campus community. None of the subjects had used the scheme previously. Four subjects were assigned to each domain.

At the beginning of the experiment, each subject was asked to study annotation guidelines which define the form-based dialog structure representation and also provides examples of dialog structure annotations in two task-oriented domains (bus schedule enquiry and UAV flight simulation; neither were used in the experiment). The material in the guidelines is quite similar to the one in Section 3.3 and Section 0. The subject was allowed to ask clarification questions about the guidelines. After the subject studied the annotation guidelines, he/she was asked to complete an annotation exercise in order to assess his/her understandability on the form-based dialog structure annotation scheme. Only a subject who sufficiently understands the annotation scheme was entered into the experiment.

During the experiment, each subject was given dialogs in one of the domains and was asked to design a dialog structure representation for the given domain, namely to develop a tagset that would be used for annotating the dialog structures of the given dialogs. The subject was then asked to annotate these dialogs according to the tagset he/she had designed. The annotation was done using CADIXE, an XML-based annotation tool. There was no time limit on any part of the experiment. On average, a subject studied the guidelines for 30-45 minutes and spent around 1-2 hours designing a markup scheme.

To control the number of annotated instances for each dialog structure component and to keep the annotation experiment simple, dialog structure annotation was divided the into two parts: concept annotation and task/sub-task annotation. The number of dialogs that each subject had to annotation is shown in the first row of Table 4.6. The total number of components that a subject annotated for each annotation task on average is shown in the second row. All the subjects in the same group were given the same set of dialogs.

	Concept annotation		**Task/sub-task annotation**	
	Air Travel	**Map Reading**	**Air Travel**	**Map Reading**
Number of dialogs	2	2	4	2
Number of components	178.8	347.8	50.5	60.8

Table 4.6: Statistic of data collected from the annotation experiment

For cross-annotation correction, each subject had to correct the markup of all the other subjects in the same group (i.e. three subjects). The subject was instructed to correct the annotation only when it did not conform to the guidelines. More detail discussion about dialog structure annotation and the cross-annotator correction procedure can be found in Section 4.2.1 and 4.2.2 respectively.

At the end of the experiment, each participant received $10 compensation for each hour they had spent doing the experiment. A performance-based bonus of up to $20 was used to motivate the participants to complete the experiment as best as possible. The subject's performance on dialog structure design and annotation was evaluated by an expert (an annotation scheme developer) on how well the annotated dialogs conform to the form-based dialog structure annotation guidelines. The performance is quantified in terms of *annotation accuracy* which is defined as the expert's acceptability in the subject's annotation. The same cross-annotator correction procedure as the one used for assessing acceptability between two subjects was used to measure the expert's acceptability. The expert first corrected each subject original annotation; annotation accuracy was then computed by comparing the original annotation against the expert's correction. The subject's performance on annotation critique and correction was computed from the similarity between the subject's critique and the expert's critique. The subject's critique and the expert's critique for each original tag are considered similar if both of them mark the correctness of the original tag (as correct or incorrect) similarly.

4.2.5 Results and discussions

For each domain, we obtained 4 sets of original annotations along with 4 tagsets developed by the subjects. We also obtained 12 pairs of cross-correction (3 from each subjects) for each domain. Each cross-correction pair consists of an original annotation and its corrected version (corrected annotation).

Two types of analysis were conducted: qualitative and quantitative analysis. In qualitative analysis, the differences among the original tagsets developed by the subjects were examined. For each type of difference, the possible causes were identified. Quantitative analysis, on the other hand, looks at the amount of changes the subjects made to the original annotations by computing acceptability for each cross-correction pair. The results from each type of analysis are given below.

4.2.5.1 Qualitative analysis

In each domain, 4 original tagsets developed by the subjects were compared against one another. The differences in dialog structure designs from the air travel planning domain and the map reading domain can be classified into 4 categories discussed below. The differences are illustrated by diagrams along with example annotations. The following notions are used in the diagrams.

- Each box represents one annotation tag in a dialog structure design.
- An arrow indicates decomposition in a structured concept and in a task/sub-task hierarchy.
- Boxes and arrows that have the same color (light or dark) were developed by the same subject.
- Overlapping boxes represent equivalent tags developed by different subjects (two tags are considered equivalent if the subjects used them to mark the same set of instances in the given dialogs)

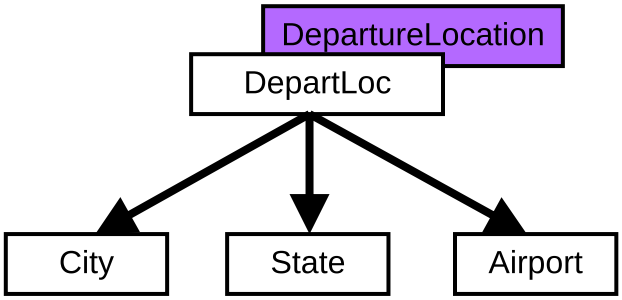
For each category of the differences, the possible causes and the corrections (if any) that the subjects made when they encountered the difference (between their own dialog structure design and the dialog structure they critiqued) are also discussed.

Differences in concept annotation

1. Structured concept granularity

   Examples

   A course-grained vs. a fine-grained annotation of "houston texas" in the air travel planning domain



- <DepartureLocation>houston texas</DepartureLocation>
- <DepartLoc>

  <City>houston</City>

  <State>texas</State>

  </DepartLoc>

   Possible causes

- The form-based dialog structure representation does not specify the appropriate level of granularity of a structured concept. The design decision may depend on an annotator's expectation on the capability of a natural language understanding module that will process the concept.

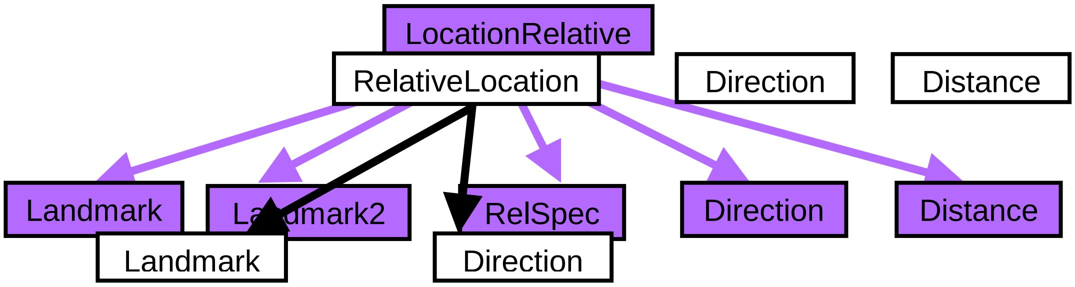
   Corrections

   Finer-grained concept structures were acceptable. A correction was made when a necessary component was missing.

2. Structured concept decomposition

Examples

Two annotations of "five inches below the gold mine" in the map reading domain with different structured concept decompositions (a **Distance** is a component of a structured concept **LocationRelative** in the first annotation but is not a component of a structured concept **RelativeLocation** in the second annotation)



- <LocationRelative>

  <Distance>five inches</Distance>

  <RelSpec>below</RelSpec>

  <Landmark>the gold mine</Landmark>

  </LocationRelative>
- <Distance>five inches</Distance>

  <RelativeLocation>

  <Direction>below</ Direction >

  <Landmark>the gold mine</Landmark>

  </RelativeLocation >

Possible causes

- The form-based dialog structure representation only specifies the definition of a structured concept; it is up to an annotator to decide on the complexity of each structured concept and a list of its components.

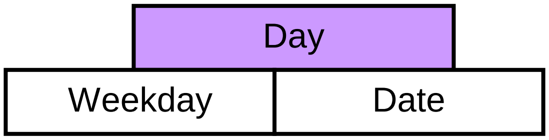
Corrections

- Additional structured concept components were acceptable
- Some missing components were added

3. Distinction between similar concepts (that have different functionalities)

   <u>Examples</u>

   Annotations of “wednesday the eleventh” in the air travel planning domain with and without the distinction between **Weekday** and **Date**



   - <Day>wednesday the eleventh</Day>
   - <Weekday>wednesday</Weekday> <Date>the eleventh</Date>

   <u>Possible causes</u>

   - A design decision, whether to differentiate between rather similar concept types (between **Weekday** and **Date** in this example) that may have different functionalities in a dialog system

   <u>Corrections</u>

   No distinction was mostly unacceptable

4. Infrequent concept annotation

   An infrequent concept is a concept that occurs only a couple of times in a given set of dialogs. Examples of infrequent concepts in the air travel planning domain are a **PeriodHold** and a **PlaneType**.

   <u>Possible causes</u>

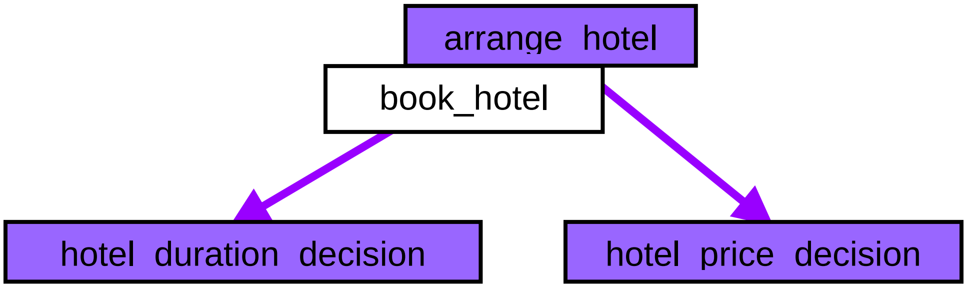
   There is not enough data for making consistent decision. We can also observe inconsistency within the same subject annotation. This might be an indication that data sparseness is also a problem for humans, not only for machine learning algorithms.

   <u>Corrections</u>

   Subjects made changes to only a few differences. One possible explanation is that the subjects did not have sufficient evidence to decide on either of the variations.

In summary, when there were differences between the subjects’ own dialog structure design and the design they critiqued, additional concept types were usually acceptable, but missing types were not. The frequency of a concept also affects the decision to make a correction.

Differences in task and sub-task annotation



1. Sub-task granularity

   Examples

   A coarse-grained sub-task vs. fine-grained sub-tasks in the air travel planning domain

   Possible causes

   - An action that associated with a low-level sub-task, for instance, **hotel_price_decision**, is correlated with the implementation of a target dialog system. Therefore, the design of a low-level sub-task is more subjective, and thus likely to be different.

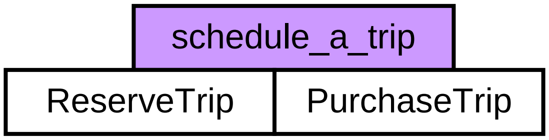
   Corrections

   - A course-grained sub-task decomposition was more acceptable than a fine-grained sub-task decomposition.
   - Additional low-level sub-tasks were considered as extra components (not corresponding to any action) and were removed

2. Distinction among different types of tasks or sub-tasks

   Examples

   Two distinct tasks (**ReserveTrip** and **PurchaseTrip**) vs. one general task (**schedule_a_trip**) in the air travel planning domain



Possible causes

- Different levels of generalization of tasks or sub-tasks and their associated forms

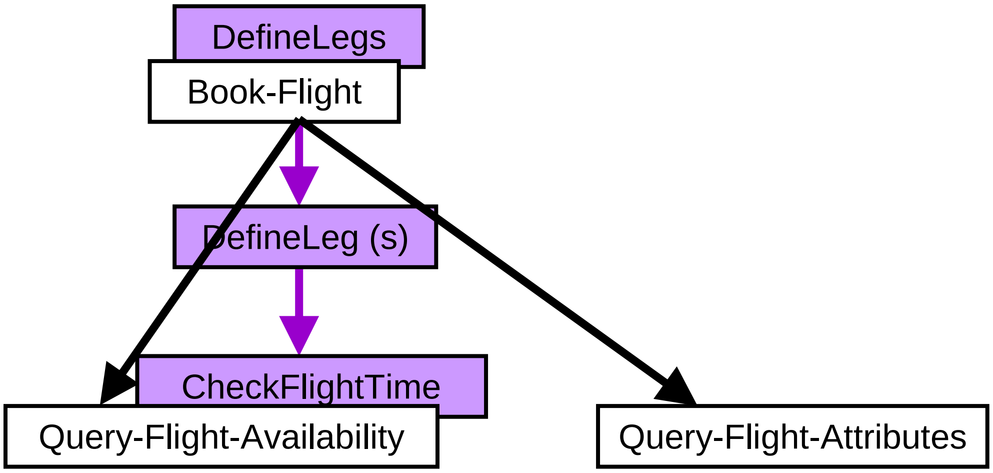
Corrections

- Extra distinction was mostly acceptable

3. Task and sub-task decomposition

Examples

Whether to have an additional sub-task (e.g. **DefineLeg**) that groups together all of the database queries made for one leg of the trip



Possible causes

- Different design decisions on how to organize related sub-tasks (into a hierarchical structure or a flat structure). These decisions may depend on how dependencies among sub-tasks are perceived. There are more dependencies between sub-tasks in the same decomposition level than between sub-tasks in different decomposition levels that do not subsume one another.
- The guidelines only discuss dependencies between top-level sub-tasks as they are decomposed from a task, but do not discuss dependencies beyond the top level sub-tasks

Corrections

- Not acceptable by some subjects

4. Missing/extra sub-tasks

   Examples

   Whether ***send_itinerary*** is an action in the air travel planning domain.

   Possible causes

   - Different decisions on what should be an action

   Corrections

   - Some extra sub-tasks were considered incorrect

In summary, when there were differences between the subjects' own dialog structure design and the design they critiqued, additional sub-tasks were usually considered as extra components (not associated with any action) and were removed. This is opposite to the correction in concept annotation. One possible reason is that the guidelines state explicitly what should not be considered as a task or a sub-task, but do not do so for a concept. Moreover, extra concepts usually provide more information for the execution of a form.

Another interesting finding is that all of the corrections on task and sub-task designs made the designs more similar to the correctors' own designs. The subjects seem to have a stronger opinion on what should be a correct design (tagset) for tasks and sub-tasks than for concepts.

4.2.5.2 Quantitative analysis

In each domain, 12 pairs of cross-correction (3 pairs from each subject) were collected. Each cross-correction pair consists of an original annotation and its corrected version (corrected annotation). Acceptability for each cross-correction pair was computed according to Equation (4.2).

I will first present the acceptability of concept annotation and the acceptability of task and sub-task annotation. Then, I will discuss annotation *accuracy* which is an expert's acceptability in a subject's annotation. Confidence in cross-annotator correction will also be discussed. Finally, the result presented in this section will be compared to other works.

Acceptability of concept annotation

The average acceptability in concept annotation of all cross-correction pairs is shown in the first row of Table 4.7. Between two types of changes, major and minor, discussed in Section 4.2.3, I am more interested in major changes as these changes reflect disagreement on dialog structure designs, which may stem from differences in the interpretation of the form-based dialog structure definition, while minor changes reflect disagreement at the instance level, which are usually caused by inconsistency in annotation. The average acceptability when all of the minor changes are excluded from the corrected annotations is shown in the 2nd row of Table 4.7. This is equivalent to using a *labeled match and minor change* criterion as a matching criterion in stead of a *labeled match* (or an *exact match*) criterion used for computing the acceptability shown in the first row. Acceptability with minor changes excluded can be considered as the acceptability of the dialog structure designs.

Reliability Measure	Air Travel	Map Reading
acceptability	0.81	0.85
acceptability – minor	0.96	0.95

Table 4.7: The acceptability of concept annotation

In both domains, the acceptability of concept annotations is quite high (> 0.8) and is high (> 0.95) when minor changes are excluded. Only 22% of the changes in the air travel planning domain are major changes while the number is slightly higher (27%) in the map reading domain. Table 4.8 shows the percentage of each error type calculated from the total number of errors. Even though the overall number of major errors is lower in the air travel domain, the percentage of Missing Type errors is higher. Since the average number of concept types that each subject designed is higher in the air travel domain than in the map reading domain (36.0 vs. 12.3), it is more likely to miss some concept types. Examples of major changes (errors) in concept annotation are discussed in detail in Section 4.2.5.1

	Air Travel	**Map Reading**
Missing Type	10.6%	4.4%
Extra Type	0.3%	6.1%
Incorrect Granularity	6.4%	7.2%
No Distinction	4.5%	9.6%
Extra Distinction	0.3%	0.0%
major errors	**22.1%**	**27.2%**
Missing Instance	17.1%	26.6%
Incorrect Label	35.0%	32.2%
Incorrect Boundary	25.8%	14.0%
minor errors	**77.9%**	**72.8%**

Table 4.8: Errors in concept annotation

Acceptability of task and sub-task annotation

For task and sub-task annotation, the acceptability is moderate in both domains as shown in Table 4.9. But if minor changes are excluded, the acceptability is quite high (> 0.8). In both domains, most of major changes come from differences in sub-task decomposition granularity. The subjects agreed on the tasks and top-level sub-tasks, but did not quite agree on low-level sub-tasks. The low-level sub-tasks are correlated with the implementation of a target dialog system; therefore, the design of the low-level sub-tasks is more subjective, and thus likely to be different. Additional low-level sub-tasks were usually considered extra components (not corresponding to any action). Therefore, a coarse-grained decomposition is more acceptable than a fine-grained decomposition. Major changes in task and sub-task annotation are discussed in detail in Section 4.2.5.1

Reliability Measure	**Air Travel**	**Map Reading**
acceptability	0.71	0.60
acceptability - minor	0.81	0.84

Table 4.9: The acceptability of task and sub-task annotation

Table 4.10 shows the percentage of each error type calculated from the total number of errors. The air travel domain has higher percent of major errors than the map reading domain since the structure of task and sub-tasks in the air travel domain is more complicated. On the other hand, the map reading domain has higher disagreement at the instance level especially on the boundaries of **draw_a_segment** sub-tasks. It is more difficult to identify a boundary between two sub-tasks of the same type than doing so for different types of sub-tasks as in the air travel domain.

	Air Travel	**Map Reading**
Missing Type	27.1%	11.2%
Extra Type	29.5%	6.3%
Incorrect Granularity	6.8%	11.2%
No Distinction	0.0%	0.0%
Incorrect Decomposition	0.0%	5.6%
Extra Distinction	1.5%	0.0%
major errors	**64.9%**	**34.3%**
Missing Instance	20.3%	37.5%
Incorrect Label	0.9%	3.7%
Incorrect Boundary	13.8%	24.5%
minor errors	**35.1%**	**65.7%**

Table 4.10: Errors in task and sub-task annotation

When comparing between annotation tasks, the acceptability of task and sub-task annotation is lower than the acceptability of concept annotation. The percentage of major errors in task and sub-task annotation is also higher. A concept is easier to observe from the transcription as its unit is smaller than a task or a sub-task. Moreover, dialog participants have to clearly communicate the concepts in order to execute a domain action. A task and a sub-task, on the other hand, correspond to larger dialog units, and associate with domain actions which are sometimes quite difficult to observe directly from the transcription (a grounding action, for example).

Annotation accuracy

Accuracy is a coding scheme expert's acceptability in a subject's annotation and is computed by the same cross-annotator correction process. Annotation accuracy is used to verify that the subjects did not agree on incorrect annotations. When both acceptability and accuracy are high, we can be assured that the high level of agreement is reasonable. The accuracy of concept annotation shown in Table 4.11 is the average of all subjects. Both acceptability and accuracy are calculated when all minor changes are excluded. Since both numbers are high, we can say that high acceptability is reasonable when compared to the expert's judgment.

Reliability Measure	Air Travel	Map Reading
acceptability – minor	0.96	0.95
accuracy – minor	0.97	0.89

Table 4.11: The acceptability and the accuracy of concept annotation

The acceptability and the accuracy of task and sub-task annotation are shown in Table 4.12. The accuracy is slightly higher than the acceptability in the air travel domain, which means that on average the subjects were more critical and made slightly more changes than the expert. Nevertheless, we are assured that the acceptability of 0.81 is not biased high when compared to the expert's judgment. However, in the map reading domain, the accuracy is moderate and is lower than the acceptability since the subjects were less strict than the expert on grounding sub-task annotation. The subjects may not have a concrete definition of this sub-task as the corresponding action is difficult to observe.

Reliability Measure	Air Travel	Map Reading
acceptability – minor	0.81	0.84
accuracy – minor	0.90	0.65

Table 4.12: The acceptability and the accuracy of task and sub-task annotation

Confidence in annotation correction

During the cross-annotator correction process, the subjects also provided the level of confidence in their correction by rating each tag as *correct*, *maybe correct*, *maybe incorrect*, and *incorrect*. By taking into account the level of confidence when computing acceptability, we can identify whether they were confident when they agreed or disagreed with other annotators.

Both acceptability measures presented in Table 4.13 are computed by Equation (4.2) but with different matching criteria. The standard acceptability, which uses *exact match* or (*labeled match*) as a matching criterion, is presented in the first row of the table. The *confident acceptability*, which uses *confident match* as a matching criterion, is presented in the second row of the table. Under this criterion only *correct* tags are considered as matched whereas *maybe correct* tags are considered unacceptable. If there are many *maybe correct* tags in an annotation critique, the confident acceptability will be lower than the standard acceptability. The detail discussion about different matching criteria is given in Section 4.2.3.

Since the confident acceptability is only slightly lower than the standard acceptability in both domains, we can say that the subjects were confident when they marked dialog structure components as *correct*. When comparing between two annotation tasks, in concept annotation, the value of confident acceptability is almost the same as the value of standard acceptability while, in task and sub-task annotation, the value of confident acceptability is slightly lower. The subjects were more confident when they rated concepts as *correct* than when they made the same decision on tasks and sub-tasks.

Reliability Measure	**Air Travel**		**Map Reading**	
	Concept	**Task/sub-task**	**Concept**	**Task/sub-task**
acceptability	0.81	0.71	0.85	0.60
confident acceptability	0.80	0.68	0.84	0.58

Table 4.13: Confidence in acceptable tags

Instead of excluding all minor changes from the corrected annotations as in *acceptability – minor*, *acceptability – unconfident* excludes all uncertain changes (the changes that are marked as *maybe incorrect*) from the corrected annotations. Only the changes that are marked as *incorrect* are considered unacceptable. The acceptability – unconfident uses a *labeled match and confident change* criterion as a matching criterion in stead of a *labeled match* (or an *exact match*) criterion used for computing the standard acceptability shown in the first row of Table 4.14. If there are many *maybe incorrect* tags in an annotation critique, the acceptability – unconfident will be higher than the standard acceptability.

In Table 4.14, the values of the acceptability – unconfident are about the same as the values of the standard acceptability in the air travel domain while the values of the acceptability – unconfident are higher in the map reading domain. These results indicate the subjects were less confident when correcting the annotations in the map reading domain than when correcting the annotations in the air travel domain. However, many unconfident corrections in the map reading domain were caused by specific coder-corrector pairs rather than by particular types of errors. When comparing between annotation tasks, the subject were less confident when correcting tasks and sub-tasks than when correcting concepts.

Reliability Measure	**Air Travel**		**Map Reading**	
	Concept	**Task/sub-task**	**Concept**	**Task/sub-task**
acceptability	0.81	0.71	0.85	0.60
acceptability – unconfident	0.83	0.75	0.91	0.70

Table 4.14: Confidence in all changes

Similar results when considering only major changes are shown in Table 4.15. *Acceptability – (minor or unconfident)* excludes all of the changes from the corrected annotations except the confident major changes. This acceptability measure uses the most relaxed matching criterion by allowing *all but confident major changes* to be considered as matched. If there are many major errors that are marked as *maybe incorrect* in an annotation critique, the acceptability – (minor or unconfident) will be higher than the acceptability - minor. In most of the cases, the subjects were confident when they made major changes as acceptability values do not change much when uncertain changes are excluded.

Reliability Measure	**Air Travel**		**Map Reading**	
	Concept	**Task/ sub-task**	**Concept**	**Task/ sub-tak**
acceptability – minor	0.96	0.81	0.95	0.84
acceptability – (minor or unconfident)	0.96	0.83	0.97	0.88

Table 4.15: Confidence in major changes

In summary, the subjects were confident when they marked dialog structure components as correct and when they made major changes to the original annotations. Some subjects were, however, not so confident when they made minor changes to the original annotations of some other subjects, but were more confident when they made major changes.

Comparison with other works

First, I would like to emphasize the difference between the evaluation procedure described in this section and a conventional procedure used in other dialog structure annotation evaluations. The goal of the human annotation experiment described in this section is to verify that the proposed form-based dialog structure framework can be understood by annotators other than a coding scheme developer and that they can generate a reasonable domain-specific tagset for a given task-oriented domain from the definitions of task, sub-task, and concept provided by the framework. I was not intending to evaluate agreement on a specific tagset for a specific domain as normally done in a conventional evaluation, which compares two annotated dialogs that are based on the same (given) tagset.

Carletta et al. (1997) applied an alternate representation to the same HCRC Map Task corpus. However, since they used a different metric to evaluate the reliability of their representation, a direct comparison cannot be made between their result and the results presented in this section. The comparison in terms of how both representations, the form-based dialog structure representation and Carletta et al.'s (1997) representation, model a dialog in the same map reading domain are provided in Section 3.4.

Other researchers who used the similar evaluation metric as the one used in this section, (Brants, 2000; Civit et al., 2003) reported bracketed precisions in the range of 0.63 - 0.92 on sentence structure annotation. The acceptability values, which are also based on bracketed precision, reported in this section are in a comparable range.

4.2.6 Conclusion

The form-based dialog structure annotation scheme can be understood and applied reliably by non-expert coders producing high acceptability in cross-annotator correction on dialog structure designs (bracketed precision > 0.8) in two disparate task-oriented domains: air travel planning (an information-access task) and map reading (a problem-solving task). High acceptability is also reasonable when compared to an expert's judgment measured in terms of annotation accuracy. Among the components in the form-based dialog structure representation, concepts can be identified more reliably than tasks and sub-tasks in both domains. High annotation scheme reliability demonstrated by these results reveals that the form-based annotation scheme is concrete and unambiguous which could also imply *learnability*.

The subjects were confident when they agreed with other subject annotations. They were sometimes not so confident when they made changes to the original annotations but mostly on minor changes. The major changes in the dialog structure designs includes: removing extraneous sub-tasks and adding missing concept types. Some of the differences between dialog structure designs were acceptable including: additional concept types and coarser-grained sub-task decomposition. The proposed evaluation methodology can accommodate acceptable annotation variations and thus helps to produce insights into annotation scheme designs.

Chapter 5

Concept Identification and Clustering

The goal of concept identification and clustering described in this chapter is to identify a set domain concepts in each task-oriented domain from the transcription of in-domain dialogs, and, for each concept, to identify a list of its members. For instance, given a set of dialogs in an air travel domain, we would like to discover that a set of domain concepts includes, **CityName**, **Airline** and **Date**, and, for a **CityName**, Pittsburgh and Seattle are its members. Since a list of concept types in a given domain is not pre-specified but will be explored from data, the concept identification problem is different from a classification problem, for example, named entity extraction. In the classification problem, a word or a group of words is classified as one of the predefined roles such as person and organization.

Since the members of the same domain concept are used in similar context in that particular domain, we can employ a word clustering technique that clusters words based on their similarity to group words that belong to the same domain concept together. There are many techniques for clustering similar words together. Since the goal of this thesis is to minimize human effort in acquiring domain-specific information and that a list of domain concepts is not pre-specified and will be explored from in-domain dialogs, I will focus on unsupervised techniques which allow a learning system to learn the domain concepts from un-annotated transcription. A decision to use an unsupervised learning approach instead of a supervised learning approach is discussed in more detail in Section 2.2.3. Two unsupervised clustering approaches are investigated, mutual information-based clustering and Kullback-Liebler-based clustering. Both algorithms, described in Section 5.1.1 and Section 5.1.2 respectively, are an iterative statistical clustering algorithm, but use different heuristics to determine the similarity between words or groups of words. For an iterative hierarchical clustering approach, a stopping criterion is an important parameter. Automatic stopping criteria based on the measures available during the clustering process are proposed in Section 5.1.3 for both clustering algorithms.

In additional to an unsupervised learning algorithm that explores domain concepts from unannotated transcription of in-domain dialogs, we can also employ existing knowledge sources that contain information about the language and the domain to improve learning accuracy. A knowledge-based clustering approach that utilizes semantic information stored in the WordNet lexical database is described in Section 5.2.

Fisher (1987) used an approach that differs from the ones discussed in this chapter to cluster categorical objects. Each object to be clustered is represented by a feature-value description. For example, a feature-value description for a fish is {BodyCover = "scales", HeartChamber = "two", BodyTemp = "unregulated", Fertilization = "external"}. The similarity between objects is calculated from a conditional probability of a feature-value pair and an inferred object class. Möller (1998) applied this conceptual clustering algorithm to infer a set of domain-specific dialog acts from a corpus of in-domain conversations. A set of features for each utterance consists of prosodic events, recognized words, and semantic structure. These features are automatically extracted from various knowledge sources available to a dialog system. The conceptual clustering algorithm that was used for dialog act acquisition is discussed in more detail in Section 2.2.2.1. Fisher's clustering algorithm allows an object to be described with multiple features while statistical clustering algorithms discussed in this chapter only utilize the statistics of word co-occurrences. Since it is not clear that a richer set of features in addition to context words are useful for concept words clustering, a simpler algorithm that is designed specifically for word clustering might be a better choice.

Some words in dialog transcription are not concept words and should be filtered out. Section 5.3 describes selection criteria that could be used to distinguish between concept words and non-concept words. The evaluation metrics that were used to evaluate the proposed concept identification and clustering approaches are described in Section 5.4. The experiment results are presented in Section 5.5. Finally, all the findings are concluded in Section 5.6.

5.1 Statistical clustering algorithms

A statistical clustering algorithm uses statistics calculated from the relation between a word to be clustered and its context to determine the similarity between words or group of words. Many statistical clustering algorithms rely on word co-occurrence statistics including the ones discussed in this section, mutual information-based clustering and Kullback-Liebler-based clustering. These algorithms groups words that occur in similar contexts together in the same cluster. The context is usually defined as immediately preceded words and immediately succeeded words. Statistics of other kinds of relations besides word co-occurrence can be used as well such as grammatical relations between a word to be clustered and other words in the same sentence (Lin, 1998). However, this technique requires a parser with may not work well with a spoken language.

Both mutual information-based clustering and Kullback-Liebler-based clustering are an agglomerative hierarchical clustering approach (or a bottom-up approach), but they use different heuristics to determine the similarity between words or groups of words. Both clustering algorithms iteratively merge words or clusters together in the order of their similarity which create a hierarchy of clusters (a treelike structure). The cluster at a leaf corresponds to a word in the vocabulary. The intermediate nodes that are closer to the leaves represent more specific word classes while the intermediate nodes that are closer to the root represent more general concepts. Hierarchical clustering provides us a more flexible way to understand and interpret the structure of a dialog since a concept may be broken down into several specific concepts or grouped into a more general concept, as needed.

5.1.1 Mutual information-based clustering

The mutual information-based clustering algorithm (MI-based clustering) used in this thesis is similar to the one described in (Brown et al., 1992)[4]. This algorithm defines the similarity between words or clusters based on their mutual information with adjacent words or clusters. The algorithm starts by assigning each word to its own cluster then iteratively merges clusters in a greedy way such that at each iteration the loss in average mutual information is minimized. After each merge, the original words or clusters in the transcript are replaced by a symbol of a new cluster and the related probabilities are recomputed. The merging process continues until the desired number of clusters is reached or a stopping criterion is met. A stopping criterion will be discussed in Section 5.1.3. *Average Mutual Information* (AMI) is defined by the following equation.

$$AMI = \sum_{i,j} p(i,j) \log \frac{p(i,j)}{p(i)p(j)} \tag{5.1}$$

where $p(i,j)$ is the bigram probability of cluster_i and cluster_j, i.e., the probability that a word in cluster_i precedes a word in cluster_j.

5.1.2 Kullback-Liebler-based clustering

In Kullback-Liebler-based clustering (KL-based clustering) similarity between words or clusters is determined by the Kullback-Liebler (KL) distance. The merging process is

[4] The MI-based clustering program that was used in the experiment was implemented by Rose Hoberman

similar to that of MI-based clustering except that the order of clusters that get merged is determined by KL-distance instead of AMI. Since the notion of distance is the invert of similarity, the KL-based clustering merges together words or clusters that have the least KL-distance first.

The KL-distance used in this research is similar to the one described in (Siu and Meng, 1999) and (Pargellis et al., 2001). A symmetric non-blow-up variant of the KL-distance, which is known as Jensen-Shannon divergence (Dagan et al., 1999), is applied to avoid the problem when one of the probabilities is equal to zero. The KL-distance between two probability functions p_a and p_b is given by the following equation.

$$J(p_a; p_b) = \frac{(D(p_a \parallel p_{\frac{a+b}{2}}) + D(p_b \parallel p_{\frac{a+b}{2}}))}{2} \tag{5.2}$$

where $D(p_a , p_b)$ is the conventional KL-distance.

$$D(p_a, p_b) = \sum_Y p_a(Y) \log \frac{p_a(Y)}{p_b(Y)} \tag{5.3}$$

A distance between cluster$_i$ and cluster$_j$, is the sum of the KL-distance between the left context probability (p^{left}) of the two clusters and the KL-distance between the right context probabilities (p^{right}) of the two clusters. More specifically,

$$Dist(i,j) = J(p_i^{left}, p_j^{left}) + J(p_i^{right}, p_j^{right}) \tag{5.4}$$

p^{left} and p^{right} are bigram probabilities. $p_i^{left}(v_k)$ is the probability that word v_k is found on the left of words in cluster$_i$. Similarly $p_i^{right}(v_k)$ is the probability that word v_k is found on the right of words in cluster$_i$. Specifically,

$$p_i^{left}(v_k) = p^{left}(v_k \mid cluster_i) = \frac{p(v_k, cluster_i)}{p(cluster_i)} \tag{5.5}$$

$$p_i^{right}(v_k) = p^{right}(v_k \mid cluster_i) = \frac{p(cluster_i, v_k)}{p(cluster_i)} \tag{5.6}$$

From the definitions of p^{left} and p^{right}, the sum in Equation (5.3) is the sum over all the context words v_k in the vocabulary.

Since Equation (5.5) and (5.6) treat all of the words in the same cluster as the same word token, after two clusters are merged, all of the occurrences of their members in the corpus are replaced with the same token and the related probabilities are recomputed. By replacing the original word tokens with the new token that represents the merged cluster, both the MI-based clustering algorithm and the KL-based clustering algorithm can be considered as a recursive clustering algorithm. An alternative way to measure the distance between two clusters without recalculating word statistics is to calculate the distance between the clusters directly from the KL-distances between individual members of the two clusters. Three linkage distances, single linkage, maximal (or complete linkage), and average linkage, define a distance between two clusters from the distances between their members as follows:

1. *Single linkage* (*Ls*) defines a distance between two clusters as the *minimum* KL-distance between members of the two clusters
2. *Complete linkage* (*Lc*) defines a distance between two clusters as the *maximum* KL-distance between members of the two clusters
3. *Average linkage* (*Lg*) defines a distance between two clusters as the *average* of the KL-distances between members of the two clusters

More detail discussion on linkage methods can be found in many textbooks and tutorials that describe hieratical clustering techniques such as Rasmussen's article (1992)

5.1.3 Stopping criteria

Since an iterative clustering algorithm can continue to merge similar clusters together until there is only one big cluster left, it is very important to choose an appropriate stop point. A stopping criterion determines when the iterative clustering algorithm should be terminated; the clusters obtained at that iteration are the output of the clustering algorithm. A good stopping criterion is the one that yields a good clustering result according to the metrics described in Section 5.4. Given that each clustering algorithm has a different merging characteristic, a good stopping criterion for each algorithm might be different. To be able to identify the last merging iteration automatically during the clustering process, a stopping criterion has to be based on the measures available in the process; examples are an AMI score in the MI-based clustering algorithm and a KL-distance in the KL-based clustering algorithm.

For MI-based clustering, two measures that are available during the clustering process, *log-AMIdelta* and *number-of-clusters,* were observed. Log-AMIdelta is the difference between AMI scores of successive iterations in log base. Number-of-clusters is the total number of the clusters that contain more than one word at a given iteration. Figure 5.1 shows the graph that plots the values of both measures versus the clustering iteration. Log-AMIdelta is presented in a lighter color while number-of-clusters is presented in a darker color. The diamond dots indicate stop points.

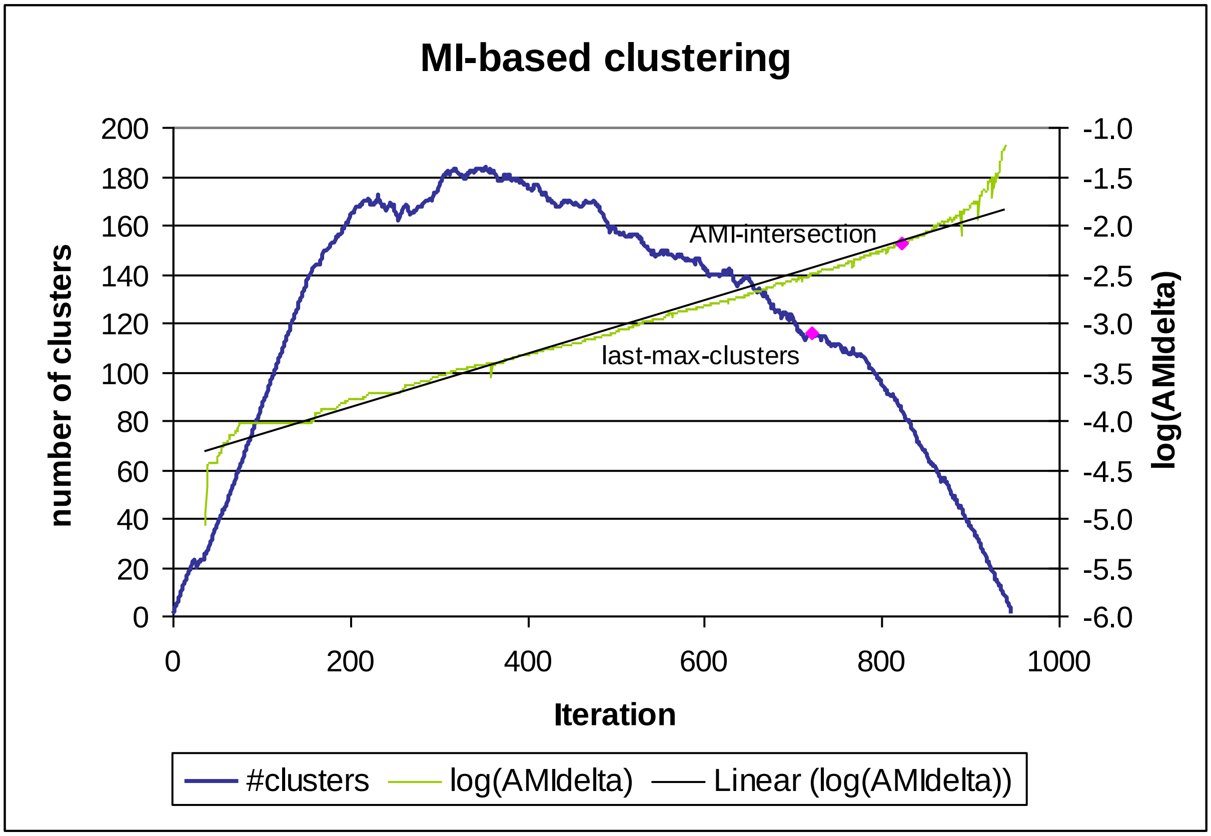


Figure 5.1: The values of two indicators, log-AMIdelta and number-of-clusters at each merging iteration of MI-based clustering

The first stopping criterion is based on AMIdelta. Since the MI-based clustering algorithm minimizes the loss in average mutual information (AMIdelta) when it merges two clusters together, AMIdelta is small at the early iteration and increases as the number of iterations increases. From the graph in Figure 5.1, we can see that the values of log-AMIdelta form a straight line for the most part but rises up at the end. It is reasonable to stop the clustering process when AMIdelta increases significantly since too much information was lost from merging two clusters together at that iteration. To obtain that iteration point, we draw a linear estimation of the values of log-AMIdelta after removing the outliers. The intersection between the line that represents the linear estimation and the one that represents the actual values of log-AMIdelta is the stop point (*AMI-intersection*).

The second stopping criterion is based on the number of clusters. The number of clusters increases when two single words are merged together, but decreases when two clusters are merged. One possible stopping criterion is the last local maximum number-of-clusters (*last-max-clusters*), which is the last iteration before the number-of-clusters decreases monotonically. This stopping criterion is justified because after this point no concept word is introduced to the clusters and irrelevant clusters may get merged together.

For KL-based clustering, two measures that can be observed during the clustering process, *KL-distance* (between the two clusters that get merged) and *number-of-clusters* (which contains more than one word), were examined. Figure 5.2 shows the graph that plots the values of both measures versus the clustering iteration. KL-distance is presented in a lighter color while number-of-clusters is presented in a darker color. The diamond dots indicate stop points.

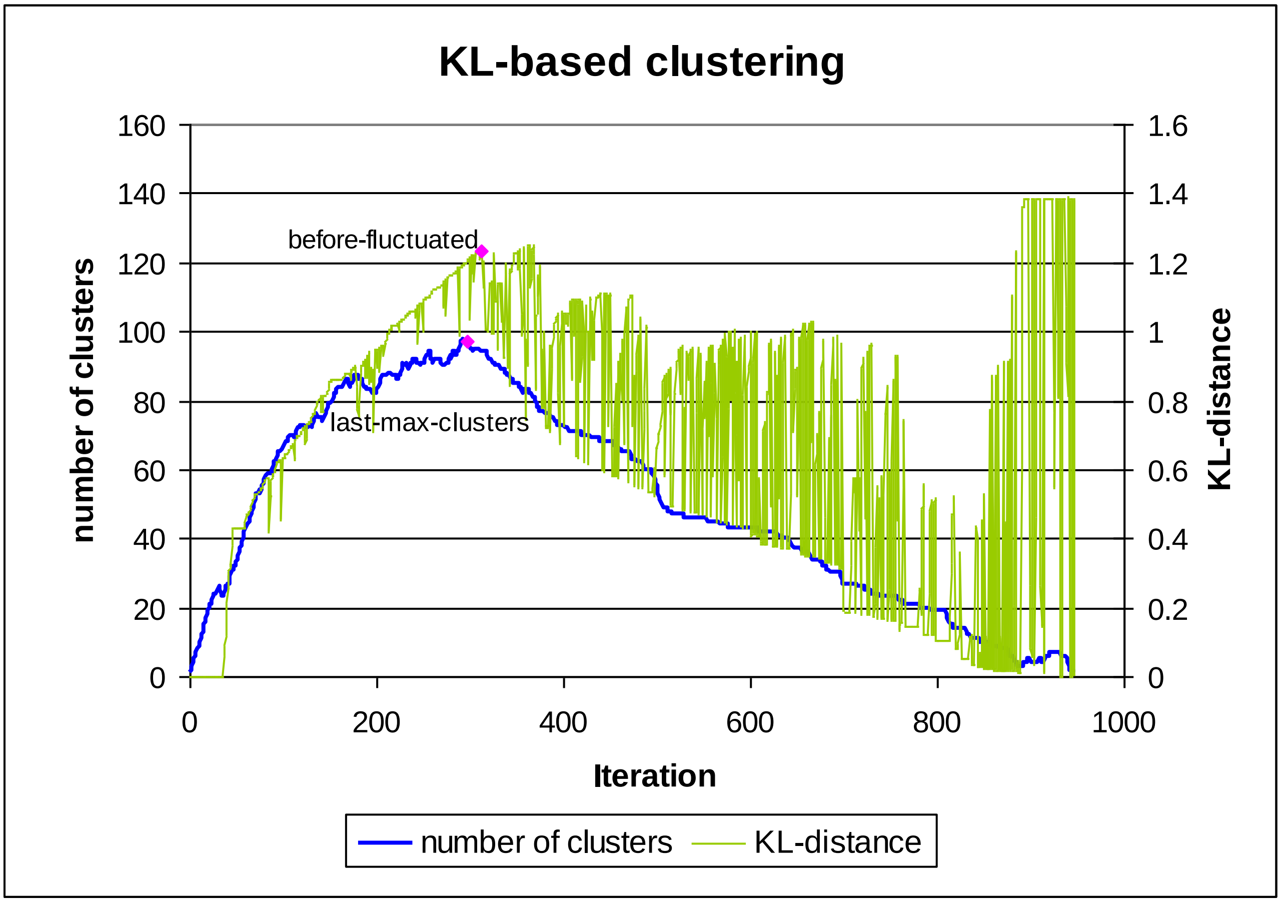


Figure 5.2: The values of two indicators, KL-distance and number-of-clusters, at each merging iteration of KL-based clustering

The first stopping criterion for KL-based clustering is based on a KL-distance. Since the KL-based clustering algorithm chooses to merge the clusters that are more similar first, the KL-distance between the two clusters that get merged at the earlier iteration is smaller. The KL-distance increases as more merging iterations are carried on as shown in the graph in Figure 5.2. However, after about 300 iterations the value of the KL-distance starts to fluctuate because many context words were already merged and were replaced by the same token. The distance value at this point may not indicate the same degree of similarity as the similar distance value at the earlier iteration. The fluctuation in the values of the KL-distance indicates that the clustering algorithm excessively merges the clusters. Therefore, the iteration that the value starts to fluctuate is a good candidate for a stop point (*before-fluctuated*). For a stopping criterion that based on the number of clusters, the same criterion as the one used in MI-based clustering, the last local maximum number-of-clusters (*last-max-clusters*), can be applied.

In the clustering algorithm that uses a linkage distance measure, a KL-distance is calculated from the original word statistic, so the distance at each merging iteration continues to increase without any fluctuation. The graph in Figure 5.3 shows a flat portion between the 250th iteration and the 350th iteration. The KL-distances at these iterations are the same and are equal to 1.386. This KL-distance value is obtained when all the left contexts of the two clusters that get merged are identical, but all their right contexts are different or vise versa. This usually occurs when the available contexts are not enough to determine the similarity. Furthermore, the identical contexts are mostly function words such as articles and preposition. For instance, “restrictions” and “base” always have the same right context “on” in this corpus. One reasonable stopping criterion is to stop right before the flat part in the KL-distance graph (*before-flatten*) since from this point onward the clusters are not quite similar or there is no enough information to determine their similarity. Riccardi and Bangalore (1998) proposed a simple stopping criterion based on a KL-distance, the median of the distance (*median*). A clustering algorithm can simply stop when the KL-distance between the two clusters that get merged exceeds the median.

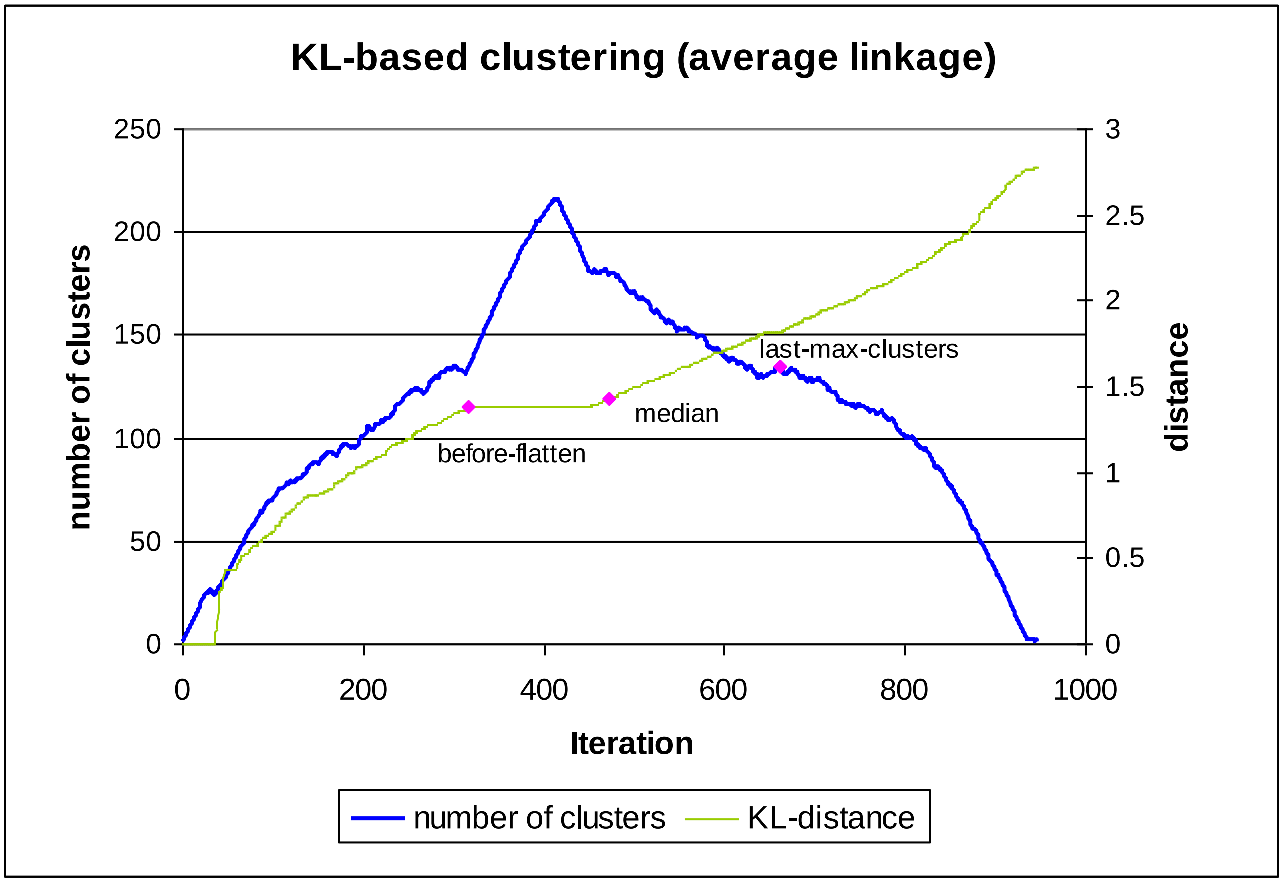


Figure 5.3: The values of two indicators, KL-distance and number-of-clusters, at each merging iteration of average linkage KL-based clustering

5.2 Knowledge-based clustering algorithms

Some domain concepts are shared among various domains and are already defined as parts of world knowledge. One example would be a concept **DayOfWeek** which consists of 7 members: “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, and “Sunday”. It seems redundant to re-discover these concepts again and again for every new domain. Furthermore, given that a statistical clustering algorithm is not perfect, it might be better to extract concepts that are domain-independent and well-defined from an existing knowledgebase instead. Since the statistical approach relies totally on data, its clustering performance is depended on statistical evidences. For example, if some concept members occur only a few times in the given data, the quality of that concept may not be so good. A concept that is obtained from a knowledgebase, on the other hand, is more accurate provided that the concept has already been defined in the given knowledge resource. Moreover, the knowledgebase does not only give us a list of members for each concept, but also provides additional information about the concept itself such as the name of the concept.

One available lexical resource is WordNet (Miller et al., 1990). WordNet is an electronic lexical database which organizes the lexicons based on their meanings rather than alphabetical order. The following properties of WordNet make it a suitable resource for concept clustering.

- It contains a rich set of relations, such as synonym, hyponym, and antonym, that connects semantically related lexicons together
- It provides detail information about each lexicon such as a gloss and its frequency
- It is freely available and so as many applications and libraries that utilize information stored in WordNet

Since each word can have multiple parts of speech and senses, in order to obtain the correct information of a specific word from WordNet, both its part of speech and word sense have to be specified. Hence, two additional steps, part of speech tagging and word sense disambiguation, are required in order to use information from WordNet for concept clustering. These two steps are discussed in more detail below.

Many automatic part of speech taggers (POS taggers) that are available have a promising performance. Nevertheless, most POS taggers were trained from a written language such as news articles. The characteristics of a written language and the characteristics of a spoken language used in task-oriented conversations are quite different. Furthermore, a set of POSs used by a tagger may be different from the one used in WordNet. To obtain the part of speech of each word in the corpus, ePost[5] tagger is used. This POS tagger is an adapted version of Brill's part-of-speech tagger (Brill, 1994) developed by Benjamin Han. ePost was trained from the Wall Street Journal corpus.

Word sense disambiguation (WSD) is a more difficult problem. Unsupervised approaches for word sense disambiguation were hardly better than the most common sense baseline according to the results of SENSEVAL-3, The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, (Snyder and Palmer, 2004). Based on this reason, ambiguous word senses are simply resolved by choosing the most common sense.

Semantic information in WordNet is organized separately for each of the four part of speech types: noun, verb, adjective and adverb. Among these four part of speech types,

[5] This POS tagger can be obtained from http://www-2.cs.cmu.edu/~benhdj/Code/index.html

noun is the one that is more likely to contain domain information. It is also a category that has been work on intensively and has a well-defined structure in WordNet. Therefore, the knowledge-based clustering will first focus on nouns. Among various semantic relations that hold between nouns, a relation that is useful for concept clustering is a *hypernym* relation or an *is-a* relation. This relation links a word to its superordinate. For example, “city” is a hypernym of “Pittsburgh” or in other words “Pittsburgh” is a “city”. Words that belong to the same type share the same hypernym. For instance, both “Pittsburgh” and “Seattle” have the same hypernym “city”. Therefore, we can group words that belong to the same type together based on their hypernyms.

After assigning a part of speech and a sense to each word in the corpus, for the word that is tagged as a noun, its hypernym is retrieved from WordNet. Then words that have the same hypernym are grouped together into one cluster.

5.3 Concept word selection

Concept word selection is a pre-processing technique that identifies words that are likely to be concept members and then passes these words to a clustering process. This pre-processing technique allows a clustering algorithm to focus only on potential concept members; nevertheless, some concept members might be filtered out. Two types of selection criteria that are based on the features of each individual word are investigated: frequency cut-off and a stop word. The first criterion is based on the assumption that concept words should occur quite often in the conversations. Hence, words that rarely occur should be filtered out. However, some words such as determiners and prepositions are very common, but do not contain any domain information. Therefore, they should not be considered as concept words. These words are known as stop words and have been widely used in the information retrieval community. The second selection criterion makes use of a stop word list. Words that are defined as stop words are filtered out. Many stop word lists are freely available; however, the differences between the characteristics of a written language and the characteristics of a spoken language have to be considered.

5.4 Evaluation metrics for concept clustering

To assess the performance of each concept clustering algorithm, the output clusters are compared directly against a set of reference concepts (created by a domain expert) in a domain of interest using the evaluation metrics proposed by Chotimongkol and Rudnicky (Chotimongkol and Rudnicky, 2002). There are two levels of evaluation metrics, a *concept-level* metric, which measures how well a clustering algorithm identifies and groups together the members of a particular concept and an *overall metric*, which measures the overall quality of all the concepts. The first step in the evaluation process is to create a mapping between the output clusters and the reference concepts. Then each cluster is evaluated against the concept that it represents using concept-level metrics. Finally, the overall quality of all the clusters is evaluated. These steps are described in detail in Section 5.4.1 – Section 5.4.3 respectively.

5.4.1 Cluster to concept mapping

Since a clustering algorithm does not assign a concept label to each cluster, a majority voting scheme is used to identify the concept that each cluster represents. Under this scheme, each word in the cluster is assigned a concept label according to the concept that it belongs to in the reference. For simplicity, each word is restricted to belong to only one concept. Then, the concept that encompasses the greatest number of words in the cluster, or the *majority concept*, is assigned as a concept label for that cluster. Based on this concept label assignment, several clusters may have the same majority concept and thus represent the same concept. A many-to-one mapping between multiple clusters and a reference concept is acceptable for concept identification. Ideally, we would like to merge every word that belongs to the same concept into a single cluster. However, a clustering algorithm may not be able to group all the members of the same concept together in one cluster. As a result, there are multiple clusters that represent the same concept. Since it is also possible to merge the clusters that represent the same concept together during a post-processing either by a human or an automatic process, allowing a many-to-one mapping between multiple clusters and a concept is better than strictly choosing only one cluster from these multiple clusters to represent the concept and missing concept members in the other clusters.

5.4.2 Concept-level metrics

Concept-level metrics indicate how well a clustering algorithm identifies and clusters the members of a particular concept. The following metrics, *precision*, *recall* and *singularity score*, are computed for each reference concept. Precision and recall measure the purity and completeness of the clusters respectively. Singularity score measures how well words that belong to the same concept are merged together. This metric was introduced by Chotimongkol and Rudnicky (2002) to address the issue of a many-to-one mapping.

A slight modification was made to the traditional precision and recall commonly used in the information retrieval community to allow multiple clusters to be compared against a reference concept. Let R_i be a reference concept of interest and C_1, C_2, …, C_{m_i} be the clusters that represent the concept R_i; where m_i is the number of the clusters. The precision and recall of the concept R_i are calculated using the following equations.

$$precision(R_i) = \frac{\sum_{j=1}^{m_i} number\,of\ words\ in\,C_j\ that\,belong\ to\,R_i}{\sum_{j=1}^{m_i} number\,of\ words\ in\,C_j} \tag{5.7}$$

$$recall(R_i) = \frac{\sum_{j=1}^{m_i} number\ of\ words\ in\,C_j\ that\,belong\ to\ R_i}{number\ of\ words\ in\,R_i} \tag{5.8}$$

Since more than one cluster is allowed to represent a concept, an additional quality metric, singularity score (SS) is used to capture how well words that belong to the same concept are merged together. When there is only one cluster that represents the concept, its singularity score gets a perfect score of 1. A penalty is imposed when a concept is split into more than one cluster. The singularity score is defined by the following equation.

$$singularity\ score(R_i) = \frac{1}{m_i} \tag{5.9}$$

To combine all three concept-level metrics into a single number, a metric called *quality score* was introduced (Chotimongkol and Rudnicky, 2002). Quality score (QS) of each concept is computed from its precision, recall and singularity score in the same way that an F-measure is calculated from precision and recall. Specifically, quality score is a harmonic mean of precision, recall and singularity score.

5.4.3 Overall metrics

In order to perform an end-to-end comparison between two clustering algorithms, a metric that indicates the overall quality of a set of output clusters is required. For each concept-level metric, its corresponding overall metric can be computed by averaging the concept-level metrics of all the concepts in a reference. For example, the overall precision can be computed by averaging the precision of all the concepts. There are two approaches for computing the average: *micro-average* and *macro-average*. Micro-average computes an average over the entire group of concepts by assigning every concept word an equal weight regardless of the concept that it belongs to. Therefore, a concept that has more members is more significant. This method is similar to the *unweighted* or *pooled* method described in (NIST, 1998). Macro-average, on the other hand, computes an average by assigning every concept in a set of reference concepts an equal weight. This method is similar to the *equal topic weighting* or *weighted* method described in (NIST, 1998). Since the number of words in each concept is not uniformly distributed, a macro-average is chosen as an averaging method to emphasize that every concept is equally important and has equal contribution toward an overall quality metric. The macro-average can be used to compute an overall metric of every concept-level metric (e.g. macro-average precision and macro-average singularity score). The quality score computed from macro-average precision, macro-average recall and macro-average singularity score provides a single number that indicates the overall quality of the output clusters.

5.5 Experiments and results

Experiments on concept identification and clustering make use of the CMU Travel Agent corpus (Eskenazi et al., 1999), which contains goal-oriented human-human dialogs between an experienced travel agent and a client arranging a trip that includes plane, hotel and car reservations. A detail discussion about the dialogs in this air travel planning domain is provided in Section 3.2. Table 5.1 shows the statistics of the CMU Travel Agent corpus.

Statistic	**Value**
Number of dialogs	39
Number of utterances	2,196
Number of an agent’s utterances (single agent)	1,108
Number of clients’ utterances (multiple clients)	1,088
Vocabulary size	947

Table 5.1: The statistics of the CMU Travel Agent corpus

A set of reference concepts in the air travel planning domain is shown in Table 5.2. This reference set contains 16 concepts with 190 concept members and was created by a domain expert. For each concept, examples of concept members and the total number of members are also given. The last column in Table 5.2 shows the number of times each concept occurs in the corpus. For simplicity, each word can belong to only one concept. Words that do not belong to any domain concept are grouped into a single *general* concept.

Concept name	Examples	Number of members	Frequency in the corpus
airline_company	Continental, Delta	15	323
airport	LaGuardia, Midway	15	101
am_pm	a.m., p.m.	2	333
area	downtown, Manhattan	11	53
car_category	compact, mid-size	2	14
car_rental_company	hertz, thrifty	6	45
car_type	automatic, manual	2	15
cardinal_number	hundred, seventy	6	153
city	Boston, Pittsburgh	44	673
date	sixth, seventh	21	225
day_of_week	Monday, Saturday	7	100
hotel_name	Hyatt, Marriott	22	97
hour_number	three, twelve	12	1188
minute_number	fifteen, thirty	12	572
month	March, April	6	109
time_period	afternoon, evening	7	172
general	want, depart	757	14411

Table 5.2: A list of concepts in the air travel planning domain

5.5.1 Statistical clustering results

Five statistical clustering algorithms were investigated: mutual information-based clustering (MI-based clustering), Kullback-Liebler-based clustering (KL-based clustering), and three variations of the Kullback-Liebler-based clustering that use a linkage distance measure (KL-based single linkage, KL-based complete linkage, and KL-based average linkage). The results are shown in Figure 5.4 - Figure 5.8 and Table 5.3 - Table 5.7 respectively. For each clustering algorithm, the corresponding graph shows the quality of the clustering result at each iteration in terms of precision, recall, singularity score (SS), and quality score (QS). These numbers are the overall metrics computed from all 16 concepts using the macro-average. The graphs in Figure 5.4 - Figure 5.8 also show the values of the measures that were used to determine the stop points. The clustering quality at the stop points are given in Table 5.3 - Table 5.7. Max-QS is an oracle stop point that yields the highest QS. The automatic stopping criteria that yield the best clustering results are highlighted in italic.

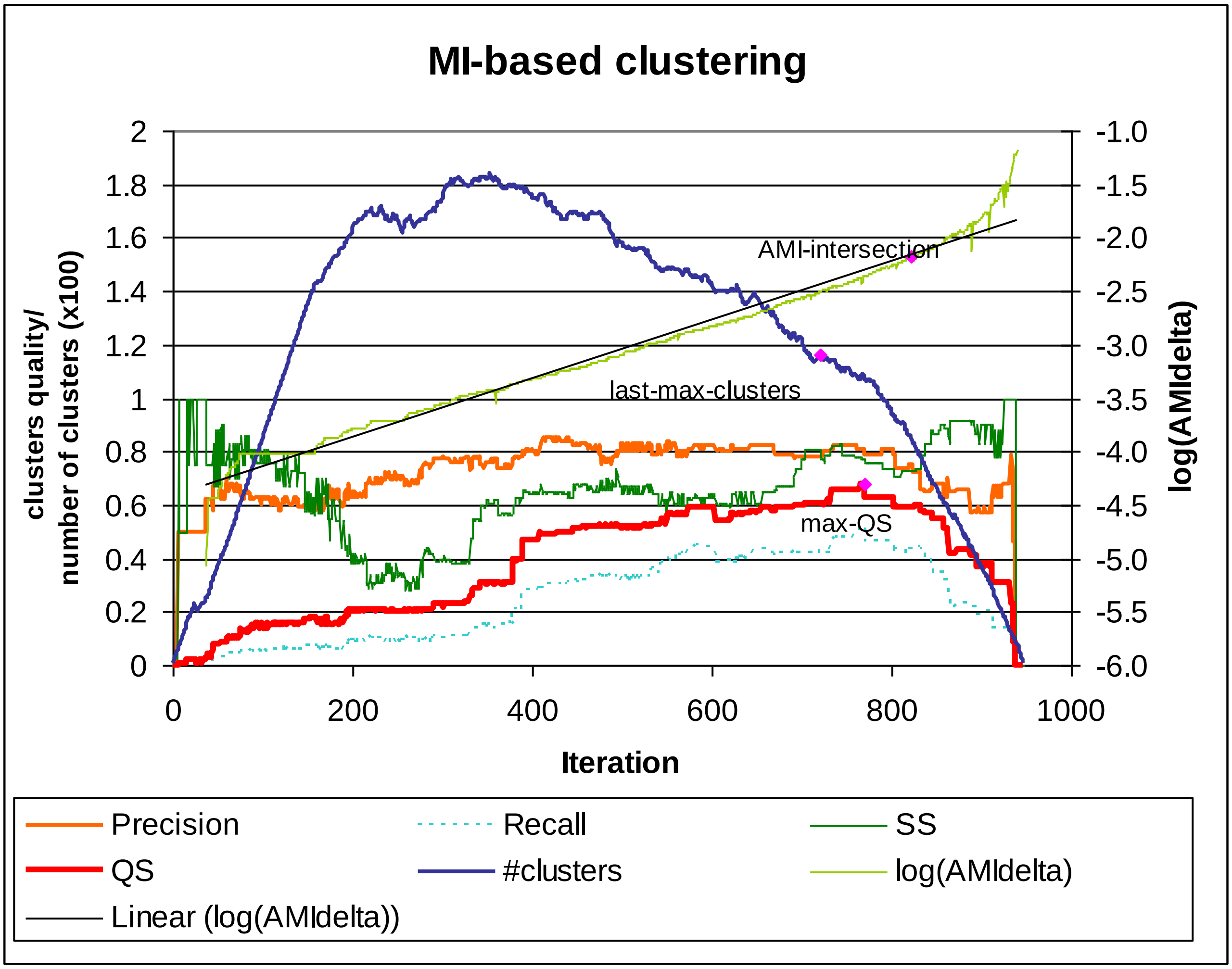


Figure 5.4: The quality of the output clusters at each iteration of the MI-based clustering

Stopping criterion	Iteration	Precision	Recall	SS	QS
Max-QS	766	0.81	0.52	0.77	0.68
AMI-intersection	815	0.74	0.42	0.73	0.59
last-max-clusters	*721*	*0.78*	*0.43*	*0.77*	*0.61*

Table 5.3: The performance of the MI-based clustering algorithm at different stopping criteria

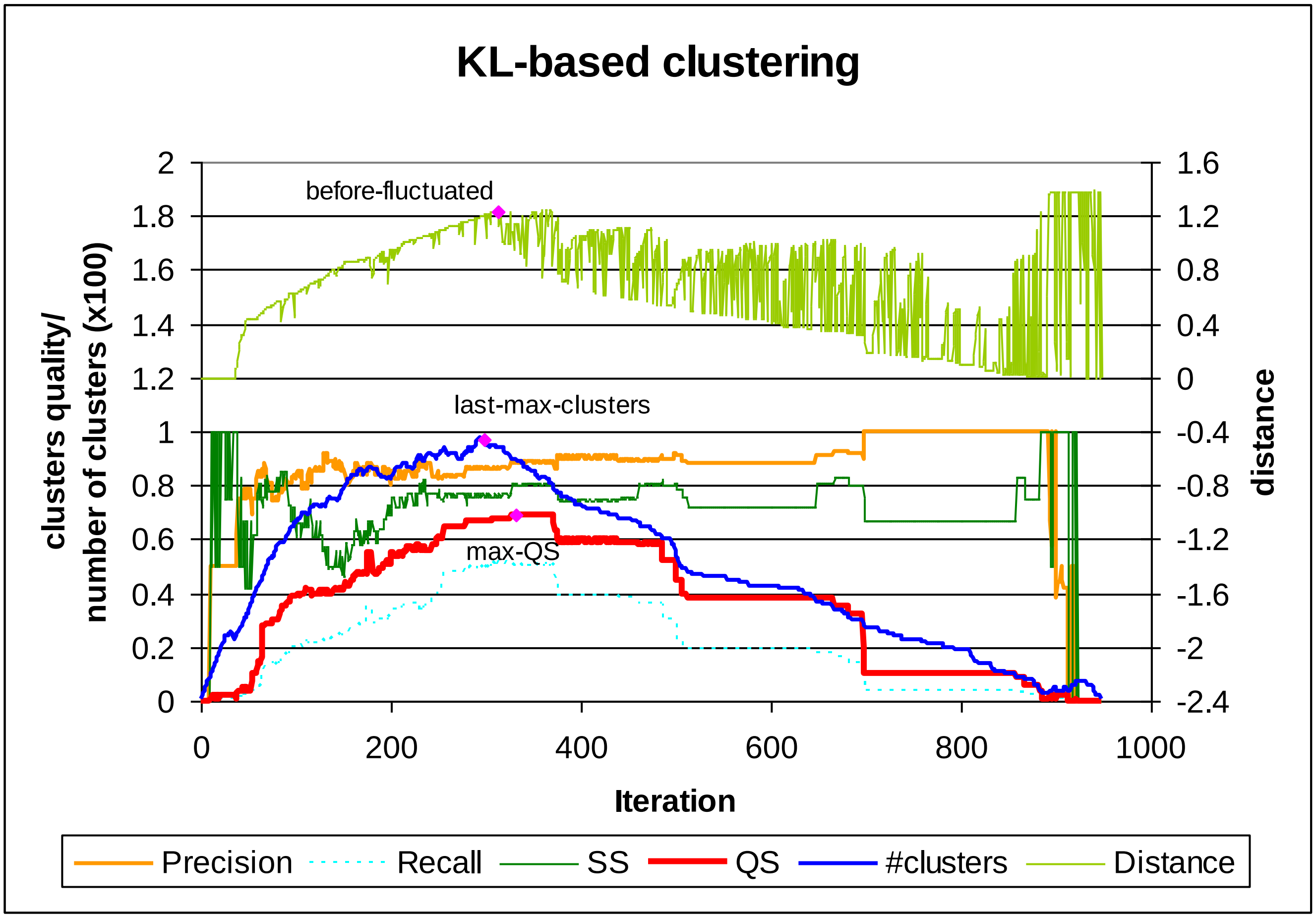


Figure 5.5: The quality of the output clusters at each iteration of the KL-based clustering

Stopping criterion	**Iteration**	**Precision**	**Recall**	**SS**	**QS**
Max-QS	327	0.89	0.50	0.81	0.69
before-fluctuated	*312*	*0.86*	*0.52*	*0.76*	*0.68*
last-max-clusters	293	0.87	0.49	0.77	0.67

Table 5.4: The performance of the KL-based clustering algorithm at different stopping criteria

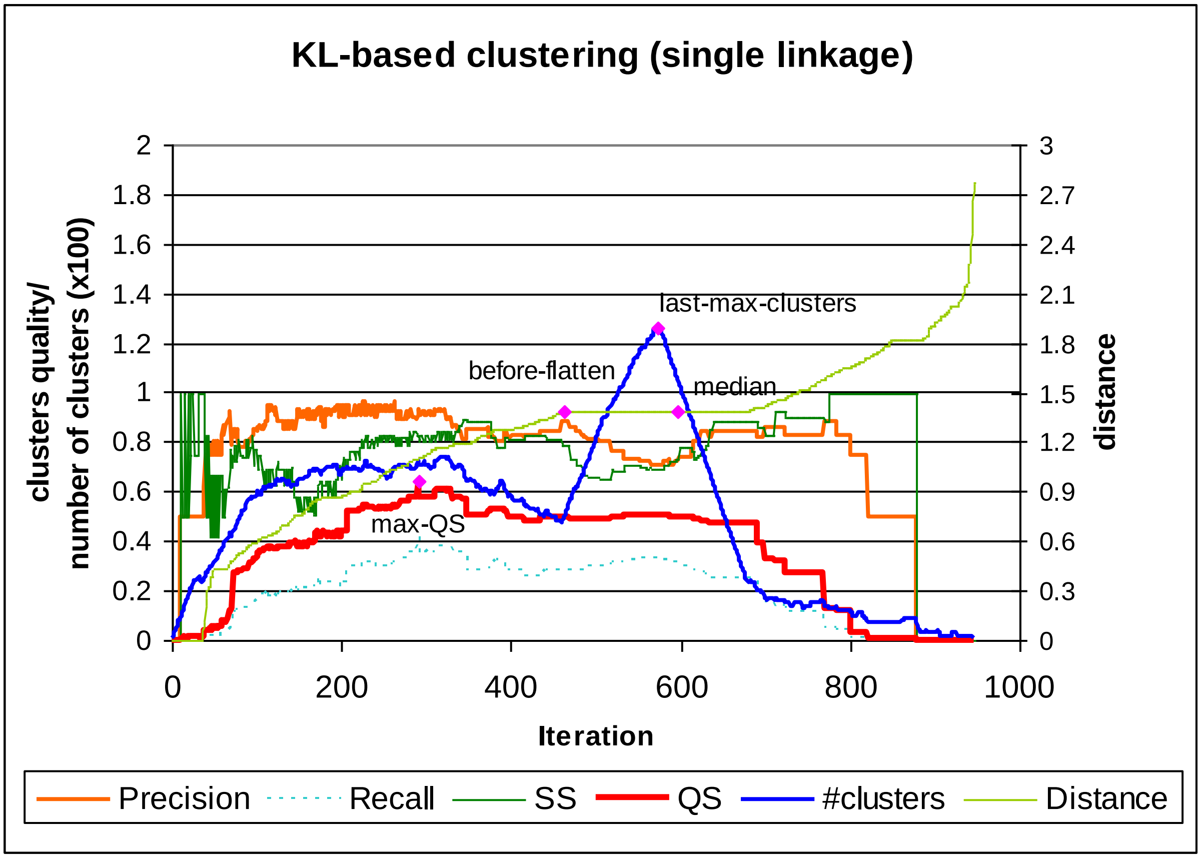


Figure 5.6: The quality of the output clusters at each iteration of the KL-based single-linkage clustering algorithm

Stopping criterion	Iteration	Precision	Recall	SS	QS
Max-QS	290	0.91	0.42	0.81	0.64
median	*569*	*0.71*	*0.33*	*0.69*	*0.51*
before-flatten	461	0.88	0.28	0.79	0.50
last-max-clusters	570	0.71	0.33	0.69	0.51

Table 5.5: The performance of the KL-based single linkage clustering algorithm at different stopping criteria

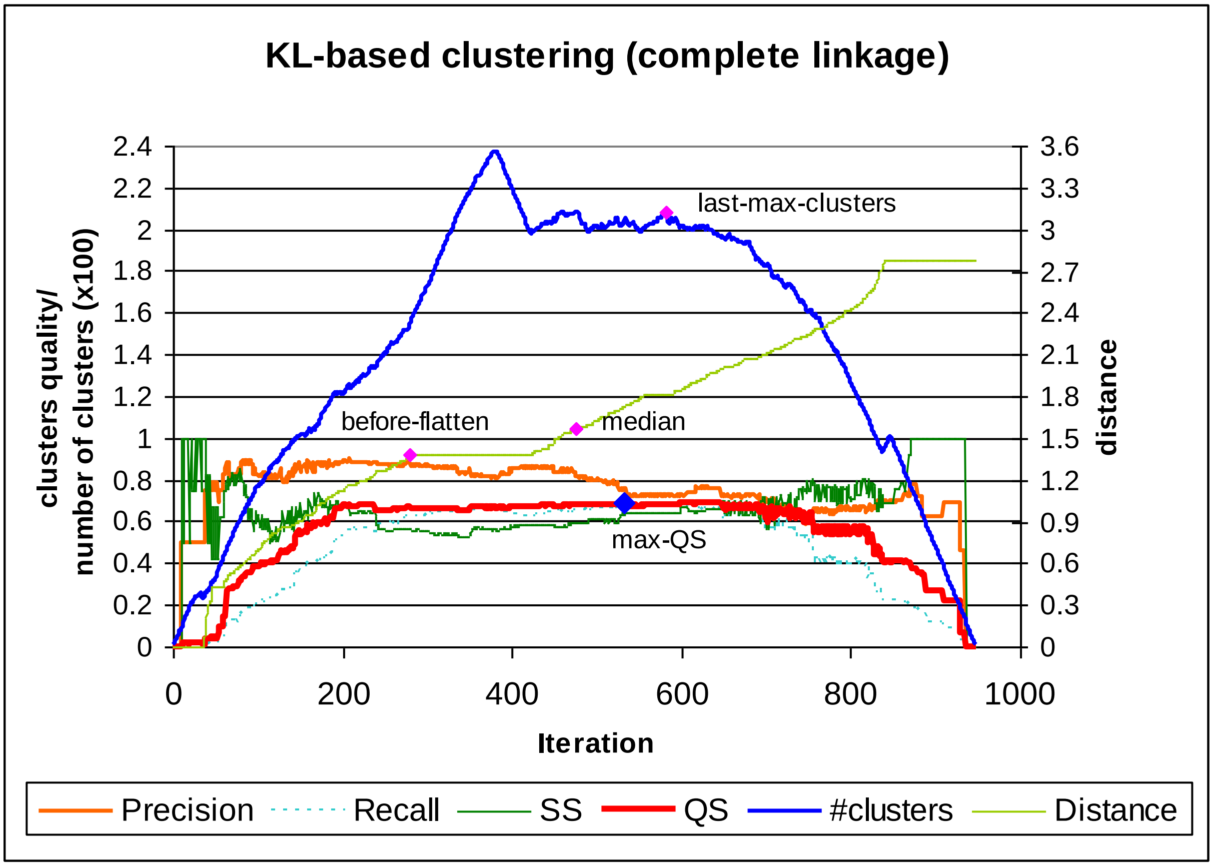


Figure 5.7: The quality of the output clusters at each iteration of the KL-based complete linkage clustering algorithm

Stopping criterion	Iteration	Precision	Recall	SS	QS
Max-QS	532	0.76	0.67	0.64	0.69
median	*474*	*0.84*	*0.65*	*0.60*	*0.68*
before-flatten	278	0.88	0.62	0.57	0.67
last-max-clusters	580	0.73	0.67	0.64	0.68

Table 5.6: The performance of the KL-based complete linkage clustering algorithm at different stopping criteria

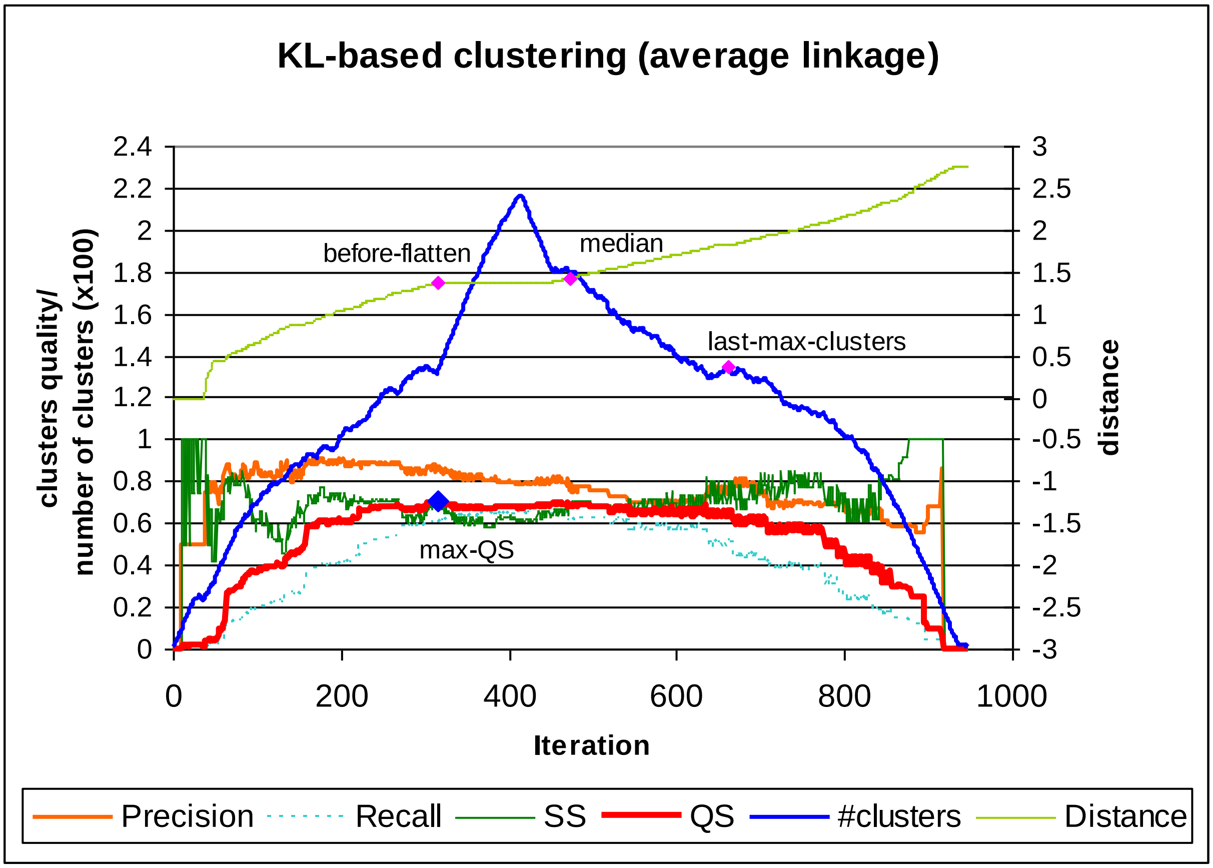


Figure 5.8: The quality of the output clusters at each iteration of the KL-based average linkage clustering algorithm

Stopping criterion	Iteration	Precision	Recall	SS	QS
Max-QS	310	0.88	0.60	0.71	0.71
median	474	0.76	0.62	0.68	0.68
before-flatten	*315*	*0.86*	*0.60*	*0.70*	*0.70*
last-max-clusters	661	0.75	0.51	0.74	0.65

Table 5.7: The performance of the KL-based average linkage clustering algorithm at different stopping criteria

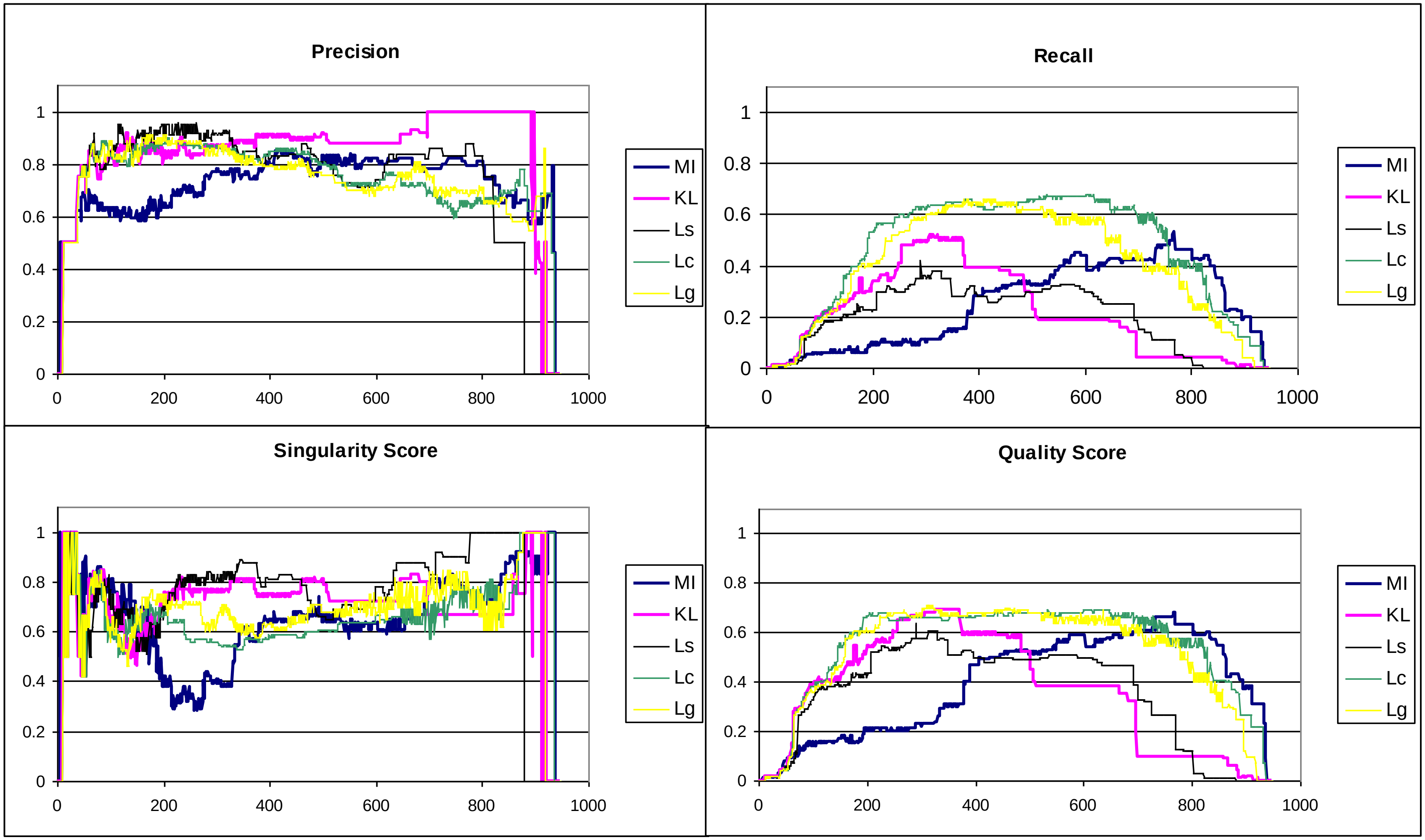


Figure 5.9: Cluster quality comparison for all statistical clustering algorithms

Stop Algorithm	Iteration	Precision	Recall	SS	QS
MI-based	766	0.81	0.52	0.77	0.68
KL-based	327	0.89	0.50	0.81	0.69
Single-linkage	290	0.91	0.42	0.81	0.64
Complete-linkage	532	0.76	0.67	0.64	0.69
Average-linkage	310	0.88	0.60	0.71	0.71

Table 5.8: The best performance of each statistical clustering algorithm

The best clustering performances of the statistical algorithms are summarized in Table 5.8. These results are obtained when the oracle stopping criterion (Max-QS) is used. Except for the KL-based single-linkage algorithm, the best quality scores of all statistical clustering algorithms are about the same. The KL-based single-linkage algorithm achieves high precision at its optimal clustering iteration; however the recall is relatively low which makes the quality score not as good as other clustering algorithms.

Algorithm	Stopping criterion	Iteration	Precision	Recall	SS	QS
MI-based	last-max-clusters	721	0.78	0.43	0.77	0.61
KL-based	before-fluctuated	312	0.86	0.52	0.76	0.68
Single linkage	median	569	0.71	0.33	0.69	0.51
Complete linkage	median	474	0.84	0.65	0.60	0.68
Average linkage	before-flatten	315	0.86	0.60	0.70	0.70

Table 5.9: The performance of each statistical clustering algorithm at its best automatic stopping criterion

The best automatic stopping criterion for each clustering algorithm is presented in Table 5.9 along with the quality of the clusters obtained at the stop iteration. For most clustering algorithms, the proposed automatic stopping criteria are able to achieve close to optimal clustering results. I would like to note that many alternative stopping criteria proposed in Section 5.1.3, although not the best criteria, also yield close to optimal clustering results as shown in Table 5.3 - Table 5.7.

For the KL-based, the KL-based complete linkage, and the KL-based average linkage clustering algorithms, the clustering results obtained from the proposed automatic stopping criteria are as good as the best clustering result obtained from the oracle stop. But, for the MI-based and the KL-based single linkage clustering algorithms, the clustering results obtained from the proposed automatic stopping criteria are quite worse than the optimal results. When examining the quality scores that all of the clustering algorithms obtained at each iteration, as shown in Figure 5.9 (bottom-right), I found that, for both the MI-based and the KL-based single linkage algorithms, there are only a few iterations that the corresponding quality scores are close to the optimal one. This might be one reason that it is difficult to find the optimal stop point automatically. For the algorithms that a near optimal clustering result can be obtained from an automatic stopping criterion (i.e. the KL-based, and especially the KL-based complete linkage and the KL-based average linkage clustering algorithm), I found that the range of iterations that have high quality scores is quite large.

From the comparison shown in Figure 5.9, I found that when the number of iteration increases, the changes in all of the quality metrics (i.e. precision, recall, singularity score, and quality score) are quite similar among all the variations of the KL-based clustering algorithm. However, these changes in the quality metrics are quite different from the ones observed from the MI-based clustering algorithm. When examining the output clusters more closely, I found that KL-based algorithms tend to merge concept words together first, and are more likely to group a single word into the exist clusters rather than create a new cluster. In contrast, the MI-based clustering algorithm tends to merge none-concept words together first, and is more likely to merge two words into a new cluster. As a result, all the quality metrics, precision, recall and singularity score, of the KL-based algorithms are higher than those of the MI-based algorithm at the early iterations. Based on this characteristic, the KL-based algorithms require fewer clustering iterations to reach the optimal clustering quality than the MI-based algorithm as shown by the QS graphs in Figure 5.9 (bottom-right). Toward the end of the clustering process, the MI-based algorithm begins to merge concept words and clusters into bigger clusters which make the values of all quality metrics increase. Based on this characteristic, the MI-based algorithm requires more clustering iterations in order to obtain the optimal or near optimal clustering result. The disadvantage of a clustering algorithm that reaches the optimal point slower is that it has more chance to merge irrelevant words into the clusters that represent the concepts, and may also merge different concepts together before it reaches the iteration that yields the optimal clustering result. The results, shown in Table 5.8, tend to agree with this analysis. The precision of the optimal clustering result achieved by the MI-based clustering algorithm is lower than the precision of the optimal result achieved by most KL-based clustering algorithms.

The characteristics of each statistical clustering algorithm are summarized in Table 5.10. The KL-based average linkage algorithm is the one that produces the best clustering result both when the oracle stopping criterion (Max-QS) is used and when the automatic stop-criterion is used.

Algorithm	**Advantage**	**Disadvantage**
MI-based	-	▪ A near optimal result can be obtained from only a few iterations ▪ Require more iterations in order to obtain a good result
KL-based (all variations)	▪ Require fewer iterations than the MI-based algorithm in order to a obtain good result	-
KL-based	-	-
Single linkage	-	▪ A near optimal result can be obtained from only a few iterations ▪ Recall is low in all iterations
Complete linkage	▪ A near optimal result can be obtained from many iterations ▪ Recall is high in most iterations	▪ Singularity scores at the stop iterations are quite low
Average linkage	▪ A near optimal result can be obtained from many iterations ▪ Achieve the highest quality score at both the oracle stopping criterion and the automatic stopping criterion	-

Table 5.10: Characteristics of each statistical clustering algorithm

Examples of the clusters obtained from the KL-based average linkage algorithm at the automatic stop point are shown in Figure 5.10. The first cluster represents **hour_number**; the second cluster represents **car_rental_company**; the third and forth clusters represent **city**. The cluster members that are marked in red and underlined are the ones that belong to other concepts. The precision of the clustering result is high as shown in the last row of Table 5.9. The first two clusters do not contain any error. For the concept **hour_number**, the clustering algorithm can identify all of its 12 concept members. The third and forth clusters contain some errors; those words should belong to the concept **airport** instead of the concept **city**. The cluster merged some airport names together with city names because they occur in quite similar context. The last two clusters also illustrate another kind of error, namely splitting. A large concept, such as **city**, sometimes gets split into multiple clusters. Singularity score, which reflects the splitting problem, is moderate for the KL-based average linkage algorithm.

Nevertheless, the recall of the KL-based average linkage algorithm is quite low. Other statistical clustering algorithms investigated in this chapter also have a similar problem. The highest recall achieved by the KL-based complete linkage algorithm is still not very high. Many of the concept words that cannot be identified are left by itself as a single word cluster rather than merged into an incorrect cluster. Some of them are infrequent concept words that the statistical approaches do not have enough evidence to merge them with any of the clusters. In other cases, some of the missing concept words have several usages. For example, "May" can be either **Month** or an auxiliary verb. Since we made an assumption that each word can belong to only one concept for simplicity, the proposed algorithms cannot handle these types of words correctly.

- TEN, TWELVE, ELEVEN, ONE, SIX, FOUR, NINE, SEVEN, FIVE, EIGHT, THREE, TWO
- HERTZ, BUDGET, THRIFTY
- GATWICK, CINCINNATI, PHILADELPHIA, L.A., ATLANTA
- LAGUARDIA, MIDWAY, MADRID, DULLES, HONOLULU, NEWARK, PITTSBURGH, SEATTLE, OTTAWA, SYRACUSE, BALTIMORE, AUSTIN, HOUSTON

Figure 5.10: Examples of the clusters obtained from the KL-based average linkage clustering algorithm

5.5.2 Knowledge-based clustering results

A resource for knowledge-based clustering is WordNet version 1.7.1. ePost tagger is used in the experiment to obtain the part of speech of each word in the corpus. The accuracy of ePost on Penn Treebank tag set, which contains 36 tags, is 90.20%. Since WordNet only distinguishes between four part of speech types (i.e. noun, verb, adjective and adverb), an accuracy on part of speech types, which is 95.60%, is a more appropriate number. It should be noted that ePost was trained on the Wall Street Journal corpus which is a written language rather than a spoken language.

After a part of speech of each word is obtained, the one that is tagged as a noun is passed to a word sense disambiguation algorithm which simply assigns the most common sense to a polysemous word. In the CMU Travel Agent corpus in which polysemous nouns account for 57.1% of the nouns, this simple WSD technique achieved 70.1% accuracy.

Clustering Algorithm	Precision	Recall	SS	QS
Average linkage	0.86	0.60	0.70	0.70
Hypernym-based with predicted nouns	0.84	0.20	0.72	0.40
Hypernym-based with all possible nouns	0.78	0.33	0.81	0.54

Table 5.11: The performance of the knowledge-based clustering approach

The performance of the hypernym-based clustering algorithm that uses predicted POSs and words senses is shown in Table 5.11. The quality of the clustering results is not good because the recall is low. One reason for the low recall is that many concept words are not tagged as nouns. For example, ordinal numbers are tagged as adjective while cardinal numbers are assigned a specially tag CD (for cardinal number). Due to this discrepancy in the definition of each part of speech type used in WordNet and the POS tagger, I decide to use the part of speech information only from WordNet instead of relying on the POS tagger. Given the vocabulary of the corpus, every word that can be a noun according to WordNet is considered. The clustering result is shown in the 2nd row of Table 5.11. Both the recall and the quality score are improved while the precision is slightly lower.

However, the recall is still low comparing to the clustering results of the statistical clustering algorithm shown in Table 5.9. The recall of the knowledge-based clustering approach is low because WordNet does not contain some domain-specific concepts especially those that are proper names such as **car_rental_company** and **hotel_name**. Nevertheless, for the concepts that are present in WordNet such as **day_of_week** and **month**, the hypernym-based clustering algorithm can identify them very accurately both in terms of precision and recall. Another problem with knowledge-based clustering is that, a knowledge resource may group concept words differently from domain-specific definition especially for abstract concepts such as time. Section 5.6 discusses some extensions that could be used to improve the clustering results for both the statistical clustering algorithm and the knowledge-based clustering algorithm.

5.5.3 Concept word selection results

Four selection criteria based on word frequency and a stop word list were experimented. The frequency cut-off was set to 2 because the corpus is quite small. The second criterion is more restrictive than the first one because it selects only words that occur at least twice in one dialog not in the entire corpus. The performance of each criterion in identifying concept words is evaluated in terms of precision and recall. The performance of each concept word selection criterion and its corresponding clustering result are shown in Table 5.12. The clustering process in this experiment utilized the KL-based clustering with average linkage since it is the best clustering algorithm as shown in Table 5.9.

	Selection Criteria	Concept Word Selection Performance				Num. clusters	Cluster Quality			
		%select	Prec.	Recall	F-1		Prec.	Recall	SS	QS
	All (baseline)	100.00%	0.20	1.00	0.33	134	0.82	0.50	0.68	0.64
1	Freq >= 2 (in a corpus)	69.42%	0.25	0.86	0.38	93	0.83	0.48	0.70	0.64
2	Freq >= 2 (in a dialog)	52.38%	**0.30**	0.79	0.44	65	0.85	0.46	0.75	0.64
3	Remove stop words	81.90%	0.24	**0.98**	0.38	113	0.82	0.49	0.69	0.64
4	Freq >= 2 (in a dialog) and remove stop words	39.89%	**0.39**	0.78	**0.52**	57	0.84	0.46	0.75	0.64

Table 5.12: The performance of each word selection criterion and its corresponding cluster quality

Among the first three criteria, using frequency cut off for within dialog frequency (2nd criterion) yielded the best precision on concept word identification while removing stop words (3rd criterion) yielded the best recall. Combining both criteria as in the forth criterion can improve both precision and F-1. Concept word selection can reduce the amount of words needs to be considered in the clustering process significantly, while the quality of the output clusters still remains the same. Even though, I expected to see an improvement on precision when non-concept words were removed, the precision was increased only slightly with a small degradation in the recall. I believe that the decrease in recall when applying a concept word selection technique can be resolved by reconsidered the filtered words in the next clustering iteration. Concept word selection also reduces the number of output clusters which makes it easier for a human to revise the clustering result.

5.6 Discussion and conclusion

Both the statistical approach and the knowledge-based approach have different strong points. For the statistical approach, no additional resource is required besides the collection of domain conversations. Since it is a data-driven method, it can capture domain-specific concepts that may not exist in a knowledgebase. Moreover, it is able to reflect domain-specific usage of some concept words such as time-related expressions and numbers. The drawback of the statistical approach is that it relies totally on the data. Therefore, if the statistical evidence is not strong enough, such as the case of a sparse data problem, the accuracy of predicted concepts may be low. Furthermore, if some concept members do not occur in the data, it is impossible to discover them. On the other hand, the knowledge-based approach can identify domain concepts very accurately providing that those concepts are in the knowledge resource. The knowledgebase provides not only information about concept members but also information about the concept itself. For example, it tells us that the cluster which contains “January”, “February”, “March”, etc., is the month of the year. Moreover, it can identify concept members that are missing from the data. While the statistical method relies heavily on the data, the knowledge-based approach is restricted to only the information available in the knowledgebase. Domain-specific words are the main drawback of the knowledge-based approach since they are not usually presented in a knowledge repository.

Recent research on automatic taxonomy induction (Snow et al., 2005, 2006) has produced efficient algorithms that can identify semantic relations, such as hypernyms and coordinate terms (words that have the same hypernym), automatically from text corpora. Promising results were reported when the relations were induced from news articles. Given a set of text from a particular domain, these data-driven approaches should be able to extend the taxonomy in WordNet to include new lexicons and their semantic relations which are specific to that particular domain, and thus increase the coverage of WordNet.

Since both the statistical approach and the knowledge-based approach have different advantages, it is better to combine both techniques together rather than choosing one. One possible combination method is to acquire an initial set of concepts through a statistical clustering approach then revise these initial concepts with a knowledge-based clustering approach. A statistical clustering approach allows us to recover as many potential concepts as possible while a knowledge-based clustering approach can improve the quality of the initial concepts by adding missing concept members and removing incorrect concept members. In addition, more efficient concept word selection criteria could be identified by adding new concept word indicators and by combining different types of criteria together. One additional type of indicator that might be worth experimenting is a name entity flag (whether a word is a name entity or not). A word that is classified as a name entity (for example, location or time expression) is likely to capture domain information.

In summary, among the five statistical clustering algorithms that were experimented (mutual information-based, Kullback-Liebler-based, and three variations of the Kullback-Liebler-based that use linkage distance measures: single linkage, complete linkage, and average linkage), the KL-based average linkage algorithm is the one that produces the best clustering result. For most statistical clustering algorithms, we are able to identify automatic stopping criteria that yield close to optimal results. A concept word selection criterion that combines word frequency cut-off with a stop word list can significantly reduce the number of words that needs to be considered in the clustering process without deteriorating the quality of the clustering result. The statistical approaches while able to capture domain-specific usage of concept words cannot accurately identify infrequent concept words due to a sparse data problem. The knowledge-based approach, on the other hand, can identify domain concepts very accurately, but on the condition that the concepts are present in the knowledge source.

Chapter 6

Form Identification

The goal of form identification is to determine different types of forms and their associated slots that a dialog system needs to know in order to perform a task in a given task-oriented domain. Since a form represents a portion of a dialog that corresponds to an action, namely a sub-task, identifying a list sub-tasks in sample dialogs can help determine a set of forms that is required in a particular domain. One approach to the form identification problem is to first segment a dialog into a sequence of sub-tasks, and then group the sub-tasks that are associated with the same form type into a cluster. By analyzing the concepts contained in each cluster, a set of slots that is associated with each form can be determined. This chapter is divided into 2 parts: Section 6.1 describes a dialog segmentation problem while Section 6.2 describes a sub-task clustering problem.

6.1 Dialog segmentation

A dialog segmentation problem can be considered as a discourse segmentation problem where a discourse unit is a sub-task in the form-based dialog structure representation. To decompose a dialog into a sequence of sub-tasks, the boundaries of the sub-tasks have to be determined. Hence, a dialog segmentation problem can also be considered as a sub-task boundary identification problem.

Segmentation algorithms for both textual and spoken data have been a subject of extensive research. Most approaches on text segmentation rely on lexical cohesion within the same topic. Lexical cohesion measures the degree of similarity within a span of text (or transcription in the case of a spoken discourse). Based on the assumption that the content within the same topic is interrelated, lack of similarity between two consecutive parts of a discourse can indicate a topic boundary. One widely-used segmentation algorithm based on lexical cohesion is the Hearst's TextTiling algorithm (Hearst, 1997). In this algorithm, lexical similarity is calculated from the cosine similarity of left and right context vectors of a candidate boundary. Potential boundaries are determined from relative changes in the similarity scores instead of from their absolute values. The candidate boundary that has a similarity score lower than both of its neighbors is considered a potential boundary. The number of boundaries (or the number of segments) in a particular discourse is determined automatically from the statistic of the similarity scores in order to reflect the characteristic of each discourse. No training data is required for the Hearst's TextTiling algorithm.

Lexical cohesion is an efficient feature for identifying a boundary between two sub-tasks that belong to different form types; however, it may not provide enough information for determining a boundary between two sub-tasks that belong to the same form type such as a boundary between two consecutive **query_flight_info** sub-tasks. The results in (Galley et al., 2003) and (Swerts and Ostendorf, 1997) confirmed this characteristic.

Another type of features that is useful for determining topic boundaries is a discourse marker. Discourse markers or discourse particles or cue phrases such as "well" and "by the way" are linguistic expressions that function as explicit indicators of structural units in a discourse. Although discourse makers are reliable features, they are not useful for a domain that contains only a few of them such as an air travel planning domain (Swerts and Ostendorf, 1997). Moreover, identifying discourse markers in a conversation may not be straight forward. Even though a list of words that can function as a discourse marker can be obtained from many literatures (Hirschberg and Litman, 1993), it is quite difficult to identify the occurrences of those words that actually convey structural information of a conversation since many of the potential markers also have alternative uses.

Instead of using a pre-defined list of discourse markers, Beeferman et al. (1999) determined a list of *domain-specific cue words* automatically from the context near the topic boundaries in the training data. Domain-specific cue words not only reflect the characteristic of each data set but also eliminate the need of a complex word sense disambiguation process that would be required in order to identify the occurrences of the pre-defined discourse markers that truly capture structural information of a discourse. A statistical framework, an exponential model, was then used to combine the domain-specific cue words and the adaptive language model probabilities which capture lexical cohesion in the discourse to determine topic boundaries in a document.

Another learning algorithm that has been applied to text segmentation is a hidden Markov model (HMM). This approach models the topics and topic shifts in the stream of text explicitly; each HMM state represents a topic while a transition between states represents a shift between topics. Each HMM state models the corresponding topic through its emission probability which can be regarded as a state-specific language model. State transition probabilities represent the probabilities of topic shifts and thus capture sequential order of topics in a document. A HMM is trained from a collection of documents to capture re-occurring patterns in an interested domain. It utilizes context similarity from multiple documents to determine topic boundaries rather than relying on lexical cohesion of the local context near each candidate boundary as in the TextTiling algorithm. To identify topic boundaries in a given document, the Viterbi algorithm is used to determine a state label for each sentence. The boundary is then predicted between any two sentences that their state labels are different.

A hidden Markov model itself is considered an unsupervised learning algorithm. However, the method used for constructing the HMM states may change the hidden Markov model to a supervised learning algorithm if the method makes use of topic boundary annotation. HMM states in the models proposed by Tür et al. (2001) and Yamron et al. (1998) were constructed by clustering similar topics in the training data together. Since this topic clustering algorithm made use of topic boundaries, those hidden Markov models are considered supervised. On the other hand, HMM states in a *content model* proposed by Barzilay and Lee (2004) were constructed by clustering similar sentences together. No topic boundary was utilized by this clustering algorithm; therefore, this hidden Markov models is considered unsupervised.

Segmentation approaches for spoken discourses can also utilize prosodic features in addition to textual features from the transcription. Correlations between prosodic features and discourse segment boundaries have been reported in many studies (Levow, 2004; Tür et al., 2001). Pitch and duration are prosodic features that are reliable across different types of spoken discourses (Levow, 2004). Unlike lexical cohesion, prosodic features are not sensitive to the content of the segment. It has been shown that prosodic features are able to identify the discourse boundaries even though the content of the segments are quite similar (Swerts and Ostendorf, 1997). Although prosodic features seem to be more robust in some cases, the algorithms that combined both textual features and prosodic features together achieved better performance than using either of them alone (Galley et al., 2003; Swerts and Ostendorf, 1997; Tür et al., 2001).

Supervised discourse segmentation approaches, such as (Beeferman et al., 1999) and (Tür et al., 2001), require training data with segment labels. However, when exploring the structure of dialogs in a new domain, as in the case of this thesis, such annotated data is not available. Therefore, we have to rely mainly on unsupervised approaches. In this section, two unsupervised discourse segmentation approaches are investigated: a TextTiling algorithm and a Hidden Markov Model. Both approaches, while performing well with expository text, require some modification when applied to spoken dialogs. One major concern is the granularity of the segments. A sub-task is rather small when compared to a topic in expository text or newscast. The topic length is 428 words in WSJ text and 996 words in CNN broadcast news (Beeferman et al., 1999) while a sub-task length is only 84 words in the air travel domain and only 55 words in the map reading domain. Since the segments that have to be identified are small, special care is required when process the context. The term “topic” and “sub-task” can be used interchangeably in this chapter as both of them refer to a discourse unit that will be identified by a segmentation algorithm.

In the following sections, I first discuss the features used in dialog segmentation and their representations. These features are used by both the TextTiling algorithm and the Hidden Markov Model. Each segmentation algorithm and the extensions that are made specifically for segmenting a dialog into a sequence of sub-tasks are described in detail in Section 6.1.2 and Section 6.1.3. The segmentation results were evaluated with three performance metrics discussed in Section 6.1.4. The experimental settings are described in Section 6.1.5 and the results obtained from the TextTiling algorithm and the Hidden Markov Model are discussed in Section 6.1.6 and Section 6.1.7 respectively. All the findings are concluded in Section 6.1.8

6.1.1 Feature representation

6.1.1.1 Word token representation

Features for dialog segmentation algorithms are taken from dialog transcription. The features include transcribed words and concept markups if a set of domain concepts has already been identified. Each transcribed word is pre-processed by removing morphological inflections using the Porter's stemming algorithm. When concept annotation is available, a concept word is represented by both a concept label and a word string. This joint representation helps disambiguating between different concept types (for instance, **[DepartureCity]:**pittsburgh is not the same token as **[ArrivalCity]:**pittsburgh) and between different word senses (for instance, “one” in “that one” is not the same token as **[hour]:**one).

6.1.1.2 Stop word treatment

One common practice in text processing is to remove stop words form a set of features as they do not carry useful information for most natural language processing applications. Conventionally, a list of stop words is prepared manually and usually includes function words such as articles and prepositions. However, this manual process is subjective and may not be optimal for every application domain. For example, numbers which are regarded as stop words in an information retrieval system, contain crucial information about date and time in the air travel planning domain. Incompatibility between a list of stop words and a target domain and genre may deteriorate the results. Gupta et al. (2006) reported a lower performance when the stop word list commonly used in the IR community was applied to a spoken dialog understanding application. Hand-turning a stop word list specifically for every application domain is inconvenient and also time consuming when working with many different domains, as in the case of natural language call router design (Kuo and Lee, 2001) and domain knowledge acquisition discussed in this thesis.

To alleviate a stop word incompatibility problem, I propose a novel approach for selecting a list of stop words specifically for each data set. In the context of a segmentation problem, a stop word is a word that carries no useful information for determining segment boundaries. Based on lexical coherence assumption used by many segmentation algorithms, words that occur regularly throughout a dialog are not informative. However, words that occur regularly are not the same as words that occur frequently. A high frequency word is still useful if its instances occur only in a specific location. For example, the word "northwest" which occurs many times in a **reserve_flight** sub-task but does not occur in a **reserve_hotel** sub-task or a **reserve_car** sub-task is undoubtedly useful for determining sub-task boundaries while the word "you" which can occur anywhere in a dialog is not so useful.

Regularity of a particular word is determined from its distribution over the course of a dialog rather than its frequency. Specifically, a *regularity count* of word *w*, *RC(w)*, is defined as the number of sliding context windows of size *c* that contain the word *w* in each dialog. The window is shifted by one utterance at a time. The parameter *c* is the same parameter as the context window size used in the TextTiling algorithm that will be described in Section 6.1.2.

Table 6.1 illustrates the calculation of regularity counts when *c* is set to 2. Each row in the table is equivalent to an utterance. "A" and "F" are two word tokens that have the same frequency; both of them occur 4 times in this text. The first column contains 8 blocks that correspond to 8 sliding context windows of size 2. The block which marked as "yes" indicates that its corresponding context window contains the word "A". The third column represents similar information but for the word "F". The number of blocks that marked as "yes" in the first column (i.e. the number of context windows contain the word "A") is the regularity count of "A", or *RC*("A"). The regularity counts of both words are shown in the last row. Since "F" is distributed more uniformly across the text, it has a higher regularity count than "A".

<table>
<tr><th colspan="2">Window contain A?</th><th colspan="6">Word token</th><th colspan="2">Windows contain F?</th></tr>
<tr><td rowspan="2">yes</td><td></td><td>A</td><td></td><td></td><td></td><td>E</td><td></td><td rowspan="2">yes</td><td></td></tr>
<tr><td rowspan="2">yes</td><td>A</td><td></td><td></td><td></td><td></td><td>F</td><td rowspan="2">yes</td></tr>
<tr><td rowspan="2">yes</td><td></td><td>B</td><td></td><td></td><td></td><td></td><td rowspan="2">yes</td></tr>
<tr><td rowspan="2">yes</td><td>A</td><td></td><td></td><td></td><td></td><td>F</td><td rowspan="2">yes</td></tr>
<tr><td rowspan="2"></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td rowspan="2">yes</td></tr>
<tr><td rowspan="2"></td><td></td><td>B</td><td></td><td></td><td></td><td>F</td><td rowspan="2">yes</td></tr>
<tr><td rowspan="2"></td><td></td><td></td><td>C</td><td></td><td></td><td></td><td rowspan="2"></td></tr>
<tr><td rowspan="2">yes</td><td></td><td></td><td>C</td><td></td><td>E</td><td></td><td rowspan="2">yes</td></tr>
<tr><td></td><td>A</td><td></td><td></td><td></td><td></td><td>F</td><td></td></tr>
<tr><td colspan="2">5</td><td colspan="6"><b>Regularity count</b></td><td colspan="2">7</td></tr>
</table>

Table 6.1: Regularity count calculation

In the TextTiling algorithm, where each dialog is processed separately, a regularity count is calculated specifically for each dialog. Therefore, the regularity count of the same word may not be the same in different dialogs. For a HMM-based segmentation algorithm, which utilizes information from all dialogs in a corpus to identify sub-task boundaries, a regularity count of each word is accumulated from all the dialogs.

A word with high regularity count is less useful in determining segment boundaries and could be considered as a stop word. A *dialog-specific stop word list* is a list of words that have a regularity count greater than a pre-defined threshold.

An alternative data-driven approach for handling stop words is to transform a regularity count into a *regularity weight (RW)* using the following equation:

$$RW(w) = \frac{total\_window\_count - RC(w)}{total\_window\_count} \tag{6.1}$$

;where *total_window_count* is the total number of sliding context windows
The frequency of each word is then weighed with the corresponding *RW(w)*. A word that has a higher regularity count is assigned a lower weight and thus contributed less in cosine similarity calculation. Regularity weight eliminates the need of an additional cut-off threshold parameter.

6.1.2 TextTiling algorithm

The TextTiling algorithm (Hearst, 1997) is a widely-used unsupervised segmentation algorithm which determines segment boundaries based on lexical cohesion. Based on the assumption that two chunks of text from the same topic are more similar than the ones from different topics, a significant drop in lexical similarity indicates a potential segment boundary. A dialog segmentation algorithm based on the TextTiling algorithm is as follows:

1. For every utterance boundary, compute cosine similarity between left and right context vectors. Context vectors are created from pre-processed transcription. Each dimension of the vector represents the frequency of each stemmed token in a context window of size c.
2. Smooth the similarity scores with average smoothing. A smoothed score is an average of all similarity scores within a smoothing window of size s.
3. Calculate a depth score for every valley in the similarity score plot. A depth score is the sum of the distance from the valley to the peaks on both sides. It takes into account the relative changes from both sides of the candidate point.
4. Output a candidate boundary that has a depth score higher than the cut-off threshold as a predicted boundary.

A set of parameters, including a context window size (*c*), a smoothing window size (*s*) and a cut-off threshold, has to be specified. These parameters affect the performance of the segmentation algorithm.

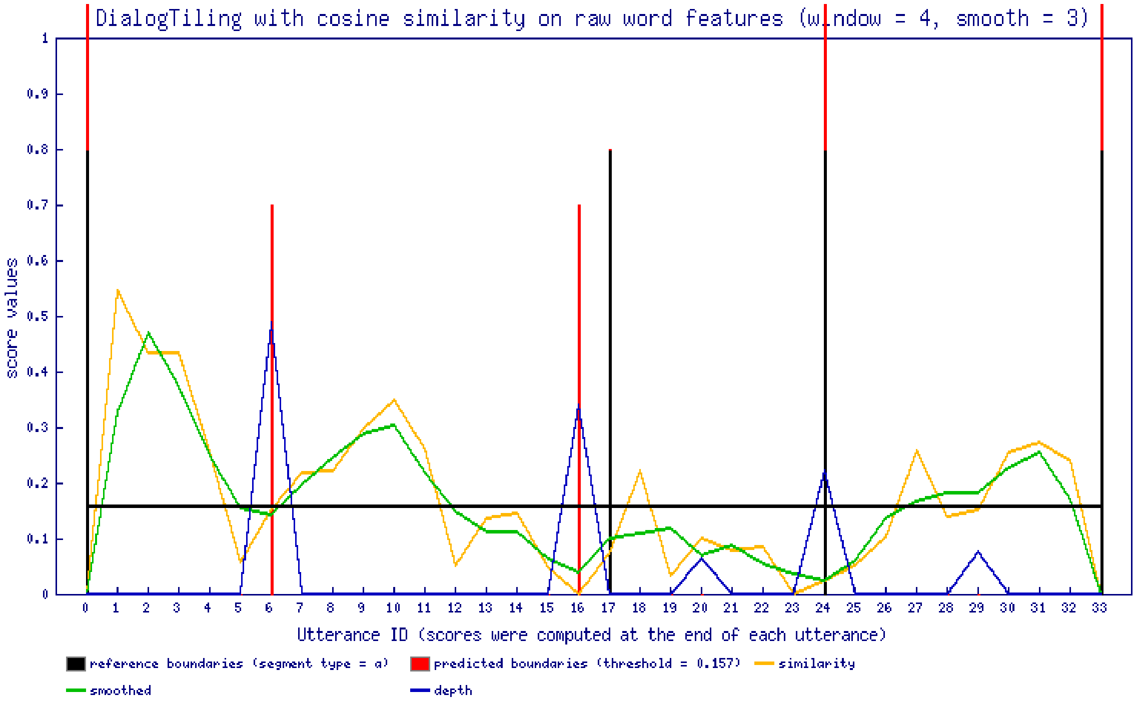


Figure 6.1: An example of a similarity score plot from the TextTiling algorithm

Figure 6.1 shows a similarity score plot produced by the TextTiling algorithm. In the plot, the x-axis represents all candidate points (utterance boundaries) and the y-axis represents similarity scores. The orange curve (lighter curve) represents cosine similarity scores while the green curve (darker curve) is the smoothed scores. The height of each blue peak is a depth score of each valley in the smoothed curve. Every peak is a potential boundary. The black horizontal line represents the cut-off threshold, which is set to $\mu - \sigma/2$; where μ is the mean of the depth scores and σ is their standard deviation. All of the potential boundaries that have depth scores greater than the cut-off threshold are selected. The boundaries identified by the algorithm are represented by red vertical lines (the shortest vertical lines). The black vertical lines (the longer vertical lines) are reference boundaries. The black-red vertical lines (the longest vertical lines) are the cases when the TextTiling algorithm predicts a boundary at the same position as a reference boundary.

In order to handle fine-grained segments in task-oriented dialogs, two types of modifications to the standard TextTiling algorithm are introduced: distance weight and triangular smoothing.

6.1.2.1 Distance weight

One crucial parameter for the TextTiling algorithm is the size of context windows. If the window size is too small, the algorithm may not have enough context to determine the similarity between two text spans that belong to the same topic and thus introduce a false alarm (false positive). On the other hand, if the window size is too large, the algorithm may find similarity between irrelevant parts of a dialog and thus miss the boundary when there is one (false negative). Since some sub-tasks are much shorter than the average length (only 2-3 utterances long), even a small context window can be considered too large for those sub-tasks.

To resolve the problem of irrelevant similarity, a *distance weight* is introduced to demote the similarity between far away contexts. Each word in the context window is weighed by the distance between the word and the considered candidate boundary. Specifically, *distance weight DW(w)* is computed as follows:

$$DW(w) = \frac{window\_size - distance}{window\_size} \tag{6.2}$$

;where *window_size* is the size of a context window (*c*), which is a parameter in the TextTiling algorithm, and *distance* is the number of utterances between the word *w* and the considered candidate boundary. A distance of a word in an utterance that is adjacent to the considered candidate boundary is 0. The distance weight calculation is illustrated below.

Figure 6.2 shows two context windows of size 4. The thick border in the middle of the figure represents a candidate boundary; the top half of the figure represents a left context window while the bottom half represents a right context window. Each row represents an utterance. The distance of "C" is 1 utterance from the candidate boundary; therefore, *DW*("C") is (4-1)/4 = 3/4. For "D", the distance is 0; therefore, *DW*("D") is (4-0)/4 = 1.

A					F
	B				
A					F
			D		
	B				F
		C			
				E	
A					F

Figure 6.2: Distance weight calculation

The distance weight is applied to each word token rather word type because distances from different word tokens to the considered boundary are not equal. For example, the distance of the first "A" in the left context window in Figure 6.2 is 3 utterances while the distance of the second "A" in the left context window is 1 utterance. Therefore, their distance weights are 1/4 and 1/2 respectively.

6.1.2.2 Smoothing algorithm

The similarity score curve needs to be smoothed in order to remove shallow valleys which represent local minima. Those local minima may interrupt the scores and create two shallow valleys instead of one deep valley that should be a boundary. The similarity score plot in Figure 6.1 is shown again in Figure 6.3. A local minimum in unsmoothed cosine similarity scores at utterance 12th creates two shallow valleys at utterance 12th and 16th instead of one deep valley at utterance 16th obtained from a smoothed curve. If the unsmoothed curve is used to identify potential boundaries, it may create a false alarm at utterance 12th if its depth score is higher than the cut-off threshold. Alternatively, the boundary at utterance 16th could be missed if the depth score of the shallower valley is lower than the cut-off threshold.

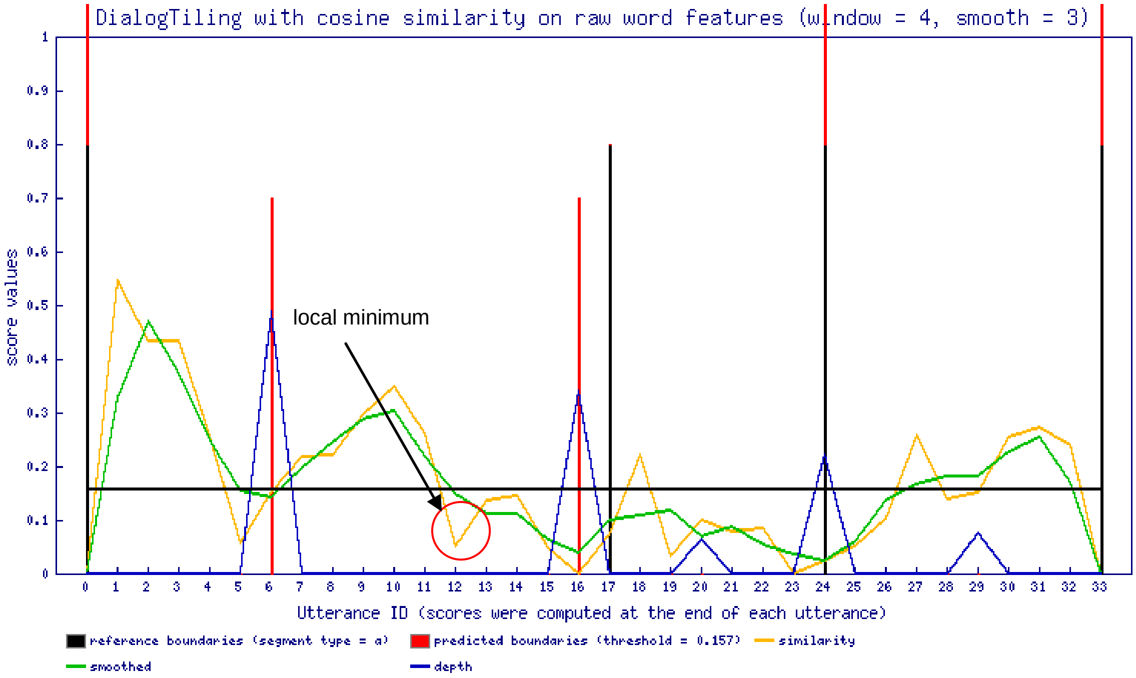


Figure 6.3: Local minimum in a similarity score plot

The original TextTiling algorithm uses a rectangular smoothing scheme which averages all of the scores in smoothing window by giving them the same weight. The following equation demonstrates how a rectangular smoothed score is computed when a smoothing window size (*s*) is set to 5.

$$smoothed\_score = \frac{X_{i-2} + X_{i-1} + X_i + X_{i+1} + X_{i+2}}{5} \tag{6.3}$$

;where X_i is a similarity score at the middle point of the smoothing window; X_{i-1} and X_{i+1} are similarity scores on the left an right of X_i respectively.

Since the sub-tasks are quite small, a precise boundary prediction is required. During a preliminary experiment, I observed that the rectangular smoothing scheme sometimes shifts the location of the valley from its original location in the unsmoothed curve and hence produced an inaccurate boundary prediction. To avoid this problem, a *triangular smoothing* scheme, which gives more weight to the scores closer to the center of the smoothing window, can be used instead. A triangular smoothed score is calculated form the following equation.

$$smoothed\_score = \frac{X_{i-2} + 2 * X_{i-1} + 3 * X_i + 2 * X_{i+1} + X_{i+2}}{9} \tag{6.4}$$

6.1.3 HMM-based segmentation algorithm

The HMM-based segmentation algorithm approaches the problem of text segmentation by modeling the topics and topic shifts in the stream of text explicitly. Each state in the hidden Markov model (HMM) represents a distinct topic while a transition between states represents a shift between topics. The text that is relevant to each topic is assumed to be generated from the corresponding HMM state according to a state-specific language model which is captured by the emission probability of that state. The shift between topics occurs according to a transition probability. The HMM-based segmentation algorithm presented in this section is adopted from Barzilay and Lee’s (2004)[6] algorithm. The process of constructing the hidden Markov model consists of two steps: HMM state induction and HMM parameter estimation.

In the HMM state induction step, HMM states, which represent topics or sub-tasks, are created automatically by clustering similar text spans together. A text span can be a sentence, a paragraph or other text stream units; nevertheless, the size of a text span may affect the granularity of induced topics and the segmentation performance. In Barzilay and Lee’s algorithm, sentences in news articles were used as text spans. Since each sentence in a news article is quite long and the notion of a topic in their evaluation tasks, information ordering and single-document summarization, is quite small, the sentence is an appropriate text span unit. However, in task-oriented dialogs, some utterances are very short and contentless. Furthermore, some utterances can occur in any sub-task. Examples of these utterances are an acknowledgement and a yes/no response. In addition, a sub-task on average is several utterances long; in the two domains that are used in the evaluation, the air travel domain and the map reading domain, the average sub-task length is 10 utterances and 7 utterances respectively. Based on these characteristics of a task-oriented dialog, a single utterance may not contain enough information that indicates its relevant topic. Therefore, an utterance unit is too small for topic induction.

Larger text spans were used in (Tür et al., 2001; Yamron et al., 1998) where HMM states were constructed by clustering similar topics together. One drawback of this approach is that it requires topic boundary in order to create the HMM states; therefore,

[6] The actual implementation is based on a Java application developed by Jaime Arguello which was used as a baseline approach in (Arguello and Rosé, 2006).

the approach is no longer unsupervised. To eliminate the need of annotated data, true topic boundaries can be approximated by predicted boundaries obtained from other segmentation algorithms such as the TextTiling algorithm. The HMM states are then induced by clustering the predicted segments together. Segmentation performances when different text span units are used to induce HMM states are discussed in Section 6.1.7.2.

A *bisecting K-means* clustering algorithm (Steinbach et al., 2000) is used to infer a set of HMM states from a set of in-domain dialogs. The bisecting *K*-means algorithm is a top-down clustering algorithm that utilizes cosine similarity between text segments in order to assign the segments into clusters. The algorithm starts with a single cluster that contains all of the text spans in the corpus. A unit of a text span can be either an utterance or a dialog segment. Then, at each iteration the largest cluster is split into two sub-clusters until the desired number of clusters is reached. This clustering algorithm is similar to the one used in sub-task clustering and is discussed in detail in Section 6.2.2. A set of clusters outputted by the bisecting *K*-means algorithm is an initial set of HMM states. The algorithm also creates an *etcetera state* by combining small clusters (i.e. the clusters that contain less than *T* text spans) together to capture the sub-dialogs that are not relevant to the task.

In the HMM parameter estimation step, two sets of HMM parameters, emission probabilities and transition probabilities, are computed. The HMM states correspond to the topic clusters created by the bisecting *K*-means algorithm, and the observations are utterances in a dialog. An emission probability of a given state captures word distribution of the corresponding topic or sub-task and can be regarded as a state-specific language model. Since the amount of dialog transcription available for training the HMM is rather small, a unigram language model is used to reduce a data sparseness problem. Specifically, the probability of an *n*-word utterance x = $w_1w_2 \dots w_n$ being generated from a state *s* can be expressed by Equation (6.5).

$$P_s(x) = \prod_{i=1}^{n} P_s(w_i) \qquad (6.5)$$

$P_s(w_i)$ is a unigram probably of a word w_i in a state s and can be estimated from its relative frequency using the following equation.

$$P_s(w) = \frac{f_s(w) + \delta_1}{\sum_{u \in V} f_s(u) + \delta_1 |V|} \qquad (6.6)$$

;where $f_s(w)$ is the frequency of w in the cluster that corresponds to a state s, V is the vocabulary and δ_1 is a smoothed count which gives a small count to words that do not occur in a state s.

A simple additive language model smoothing technique, *plus-delta*, is used to estimate both the emission probabilities and the transition probabilities (discussed below). Nevertheless, a more sophisticated smoothing technique could be investigated. Chen and Goodman (1996) discussed several smoothing techniques and then compared those techniques empirically in the field of language modeling.

For an etcetera state, a complementary language model similar to the one described in (Barzilay and Lee, 2004) is used. For a HMM that consists of m states where s_m is an etcetera state, an emission probably of an etcetera state is given by the following equation.

$$P_{etc}(w) = \frac{1 - max_{i:i<m}\ P_{s_i}(w)}{\sum_{u \in V}(1 - max_{i:i<m}\ P_{s_i}(u))} \tag{6.7}$$

A state transition probability is estimated from a relative frequency of a transition between two specific states in the training data. Each utterance in a dialog is assinged a state label that is similar to the state label of the cluster that it belongs to. Specifically, let l_t be a state label of an utterance t and $C(s_i, s_j)$ be the number of transitions from state s_i to s_j in the training data, namely, the number of state label pairs where $l_t = s_i$ and $l_{t+1} = s_j$. Equation (6.8) illustrates the smoothed estimation of the transition probability from state s_i to s_j. δ_2 is a smoothed factor.

$$P(s_j \mid s_i) = \frac{C(s_i, s_j) + \delta_2}{\sum_{i=1}^{m} C(s_i, s_j) + \delta_2 m} \tag{6.8}$$

Since the HMM states represent the sub-tasks, a boundary can be placed between two utterances that are assigned different state labels. However, the initial clustering obtained from the bisecting K-means algorithm does not take into account sub-task ordering information. To incorporate the ordering information, the Viterbi algorithm, which utilizes the ordering information through the HMM transition probabilities, is used to find the best state sequence of the observed utterances. The new state labels obtained from the Viterbi decoding are then used to re-estimate the HMM parameters using the same equations given above. The Viterbi decoding and the HMM parameter re-estimation can be iterated until converged, i.e. the number of utterances that are assigned new state labels is less than a pre-defined threshold (*threshold-2*), or the maximum number of iterations is reached.

To identify sub-task boundaries in a given dialog, the Viterbi algorithm is used to determine the best state sequence of all the utterances in the given dialog. This Viterbi algorithm is the same algorithm as the one used for constructing the HMM model. The boundary is then predicted between any two utterances that their state labels are different. Practically, since the HMM-based segmentation algorithm is an unsupervised learning approach, we can construct the HMM model from all available dialog data. In this case, sub-task boundaries in each dialog can be determined from the state labels assigned to the utterances in the last iteration of the Viterbi decoding and HMM parameter re-estimation process.

6.1.4 Evaluation metrics

To evaluate the performance of each dialog segmentation algorithm, the predicted boundaries are compared against the sub-task boundaries annotated by a coding scheme expert. Two types of evaluation metrics are used: P_k and F-measure. P_k is a probabilistic error metric proposed by Beeferman et al. (1999) that measures the probability of misclassifying two utterances that are k utterances apart as belong to the same sub-task or different sub-tasks. P_k is calculated by counting the classification errors via a moving window of length k. At each location, the algorithm determines whether the two ends of the probe are in the same or different segments in the reference segmentation, and increases a counter if the predicted segmentation disagrees. The total count is then normalized by the number of measurements taken to make the P_k value scales between 0 and 1. Since P_k is a probability of segmentation errors, a lower P_k value is preferred. An algorithm that predicts all boundaries correctly receives a score of 0. The following equations formally defined P_k.

$$P_k = P_{miss} * p(\text{different } ref \text{ segments}) + P_{FalseAlarm} * p(\text{same } ref \text{ segment}) \qquad (6.9)$$

;*p*(same *ref* segments) is the probability that two points that are k utterances apart are in the same segments in the reference while *p*(different *ref* segments) is the probability that two points that are k utterances apart are in the different segments in the reference.

$$P_{miss} = \frac{\sum_{i=1}^{N-k} \delta_{hyp}(i,i+k)*(1-\delta_{ref}(i,i+k))}{\sum_{i=1}^{N-k}(1-\delta_{ref}(i,i+k))} \tag{6.10}$$

$$P_{FalseAlarm} = \frac{\sum_{i=1}^{N-k}(1-\delta_{hyp}(i,i+k))*\delta_{ref}(i,i+k)}{\sum_{i=1}^{N-k}\delta_{ref}(i,i+k)} \tag{6.11}$$

;where the summations are over all of the utterances in each dialog and where
$\delta(i,j) =$ 1 when utterances *i* and *j* are form the same sub-task
0 otherwise

The value of P_k is also depended on the choice of *k*. Beeferman et al. (1999) suggested that the value of *k* should be set to half the average true segment length. However, there are several alternatives on how the average segment length could be computed. The first variation is the choice for the unit of *k*. Even though an utterance is used as a unit in the illustration given above, other units of discourse can be used interchangeably. In the context of dialog segmentation, the unit of *k* could be word or utterance. Additionally, an average segment length can be calculated from all of the dialogs in the corpus or calculated separately for each dialog; this introduces another variation in the calculation of *k*. Among several alternatives, an appropriate method for calculating the value of *k* is determined empirically from an experiment discussed in Section 6.1.5.2.

F-measure (or *F-1*) is the harmonic mean of precision and recall. However, the standard F-measure does not give any credit to a near miss boundary. Some modifications have been made to the standard F-measure to allow some near misses to be considered as a match. For example, a credit is given to a boundary that is within an arbitrary fixed range from the reference boundary. Since the segmentation result will later be used to identify a set of forms and their associate slots, the predicted segment that contains the same set of concepts as the reference segment is acceptable even though their boundaries are slightly different. For that reason, it is more suitable to define a close match relative to the location of the concepts inside each segment rather than defining it based on a fixed distance. For this reason, a near-miss boundary is counted as a match if there is no concept between the near-miss boundary and the closest reference boundary. This extension to the standard precision and recall are referred to as *concept-based precision* and *recall*. *Concept-based F-1* is the harmonic mean of concept-based precision and recall. The segmentation algorithm that achieves high concept-based F-1 may not produce very accurate segment boundaries, but it will; nevertheless, produce segments that contain similar sets of domain concepts as the ones in the reference segments.

6.1.5 Experimental settings

The proposed dialog segmentation algorithms were evaluated with dialogs from two task-oriented domains: the air travel planning domain and the map reading domain described in Section 3.2 and Section 3.4 respectively. Reference sub-task boundaries were annotated by a coding scheme expert. For simplicity, the annotator was only allowed to place sub-task boundaries at utterance boundaries.

6.1.5.1 Corpus statistic

The test corpora consist of 24 dialogs from the air travel planning domain and 20 dialogs from the map reading domain. The dialogs from the air travel planning domain were annotated with the task structure presented in Table 3.3 while the dialogs from the map reading domain were annotated with the task structure presented in Table 3.5. The statistics of annotated dialogs is shown in Table 6.2.

	Air Travel		**Map Task**	
	mean	**std.**	**mean**	**std.**
Dialog length (utterance)	55.6	23.0	125.5	36.5
Utterance length (word)	8.2	7.8	7.5	7.9
Number of segments per dialog	5.4	1.5	17.1	4.9
Segment length (utterance)	10.3	7.1	7.4	4.8
Segment length (word)	84.4	60.2	55.2	39.5

Table 6.2: The statistics of the test corpora

The sub-task is quite short when comparing to a topic in expository text or newscast. The topic length is 428 words in WSJ text and 996 words in CNN broadcast news (Beeferman et al., 1999) while an average sub-task length is 84 words in the air travel domain and only 55 words in the map reading domain. Furthermore, sub-task length variance is high especially in the air travel domain. Some sub-tasks in this domain are quite long due to lengthy negotiation while some sub-tasks are only 2-3 utterances long such as a **query_flights_fare** sub-task.

6.1.5.2 Defining *k* for P_k

Four degenerate segmentation algorithms similar to the ones described in (Beeferman et al., 1999) were implemented to identify an appropriate method for calculating *k*.

1. *ALL* predicts a boundary at the end of every utterance (at every candidate point)
2. *NONE* predicts no boundary at all (except for the beginning of the dialog and the end of the dialog which are default boundaries)
3. *EVEN* predicts a boundary at every m^{th} utterances, where *m* is the average reference segment length for each dialog
4. *RAND* predicts *n* boundaries randomly, where *n* is the number of reference boundaries in each dialog

Each degenerate algorithm was evaluated with 4 variations of P_k that differ along two aspects: 1) unit of *k* (utterance or word) and 2) average segment length calculation (averaging per dialog or per data set).

Algorithm	**Number of boundaries**	**Calculate *k* per dialog**			**Calculate *k* per data set**		
		False alarm probability	**Miss probability**	$\mathbf{P_k}$	**False alarm probability**	**Miss probability**	$\mathbf{P_k}$
ALL	56.625	1.000	0.000	0.578	1.000	0.000	0.550
NONE	2.000	0.000	1.000	0.422	0.000	1.000	0.450
EVEN	6.667	0.422	0.509	0.456	0.463	0.476	0.426
RAND	6.417	0.363	0.647	0.481	0.421	0.631	0.482

Table 6.3: Utterance-based P_k of degenerate algorithms in the air travel domain

Table 6.3 presents the results of the degenerate algorithms in the air travel domain when the unit of *k* is an utterance (utterance-based P_k). The results when *k* is calculated separately for each dialog and the results when *k* is calculated from the average segment length of all the dialogs in the data set are compared. When *k* is calculated per dialog, the P_k value of the NONE algorithm is much better than the P_k value of the ALL algorithm. These results are different from the ones reported in (Beeferman et al., 1999) and (Arguello and Rosé, 2006). When *k* is calculated per data set, the difference between the ALL algorithm and the NONE algorithm becomes smaller.

It can be deduced from Equation (6.9) that when the false alarm probability is equal to 1 and the miss probability is equal to 0, P_k is equal to *p*(same *ref* segments). Similarly, when the false alarm probability is equal to 0 and the miss probability is equal to 1, P_k is equal to *p*(different *ref* segments). Based on the results of the ALL and NONE algorithms, it can be deduced that *p*(same *ref* segments) is equal to 0.578 while *p*(different *ref* segments) is equal to 0.422 when *k* is calculated per dialog. One possible reason for the unbalanced probabilities is that the segment length has high variance as show in Table 6.2. When k is calculated per data set, it can be deduced that p(same ref segments*)* is equal to 0.550 while *p(*different *ref* segments*)* is equal to 0.450. These two probabilities are closer together than when *k* is calculated per dialog. This result might indicate that calculating an average segment length from all of the dialogs in the data set is a better method since it reduces segment length variance.

There is oracle information inherited in the EVEN and RAND algorithms. In the RAND algorithm, the number of boundaries in the reference is given while in the EVEN algorithm both the number of boundaries and the average sub-task length in the reference segments are given. The EVEN algorithm, which uses more oracle information, has a better P_k than the RAND algorithm. This is also different from the results reported in (Beeferman et al., 1999). One reason for this might be from the fact that in the air travel domain a span of text to be segmented, a dialog, is much shorter than continuous text corpus in the broadcast news domain. Since both pieces of oracle information are given per dialog, they have a lot of influence on the segmentation result. In the map reading domain, where an average dialog length is longer, difference in P_k values between the EVEN and the RAND algorithms is smaller. The results from the map reading domain are given in Table 6.5 and Table 6.6.

Algorithm	**Number of boundaries**	**Unit = utterance (k=5)**			**Unit = word (k=42)**		
		False alarm probability	**Miss probability**	**P_k**	**False alarm probability**	**Miss probability**	**P_k**
ALL	56.625	1.000	0.000	0.550	0.989	0.000	0.544
NONE	2.000	0.000	1.000	0.450	0.000	1.000	0.450
EVEN	6.667	0.463	0.476	0.426	0.406	0.428	0.376
RAND	6.417	0.421	0.631	0.482	0.343	0.624	0.443

Table 6.4: P_k values of degenerate algorithm when *k* is calculated from all of the dialogs in the air travel domain

The results of the degenerate algorithms in the air travel domain when a unit of *k* is an utterance (utterance-based P_k) and the results when a unit of *k* is a word (word-based P_k) are compared in Table 6.4. *k* is calculated per data set. When a unit of *k* is a word (word-based P_k) the false alarm probability of the ALL algorithm may not be equal to 1 because some utterances can be longer than *k* words. When a unit of *k* is a word, better P_k values are obtained for all degenerate algorithms. Using a word as a unit helps overcoming an utterance length variation problem.

Algorithm	**Number of boundaries**	**Calculate *k* per dialog**			**Calculate *k* per data set**		
		False alarm probability	**Miss probability**	**P_k**	**False alarm probability**	**Miss probability**	**P_k**
ALL	126.500	1.000	0.000	0.537	1.000	0.000	0.509
NONE	2.000	0.000	1.000	0.463	0.000	1.000	0.491
EVEN	18.500	0.480	0.477	0.476	0.517	0.443	0.468
RAND	18.050	0.418	0.573	0.488	0.439	0.556	0.488

Table 6.5: Utterance-based P_k of degenerate algorithms in the map reading domain

The utterance-based P_k of the degenerate algorithms in the map reading domain is shown in Table 6.5. When *k* is calculated separately for each dialog, *p(*same *ref* segments*)* is equal to 0.537 while *p(*different *ref* segments*)* is equal to 0.463. However, when *k* is calculated from all of the dialogs, these two probabilities are more balance, *p(*same *ref* segments*)* = to 0.509 while *p(*different *ref* segments*)* = 0. 491. These probabilities are closer together than those in the air travel domain as an average segment length in the map treading domain has lower variance.

Algorithm	**Number of boundaries**	**Unit = utterance (k=4)**			**Unit = word (k=28)**		
		False alarm probability	**Miss probability**	**P_k**	**False alarm probability**	**Miss probability**	**P_k**
ALL	126.500	1.000	0.000	0.509	0.953	0.000	0.504
NONE	2.000	0.000	1.000	0.491	0.000	1.000	0.470
EVEN	18.500	0.517	0.443	0.468	0.446	0.423	0.417
RAND	18.050	0.439	0.556	0.488	0.367	0.555	0.445

Table 6.6: P_k values of degenerate algorithm when *k* is calculated from all of the dialogs in the map reading domain

Similar to the air travel planning domain, better P_k values for all degenerate algorithms are obtained when using a word as a unit of *k* instead of an utterance as shown in Table 6.6.

Summary

From the results in both the air travel domain and the map reading domain, I chose to compute *k* from all dialogs in the corpus since the average segment length calculated from more data reduces segment length variance. In terms of the unit of *k*, a word unit was chosen. The finer-grained unit resolves an utterance length variation problem.

6.1.6 TextTiling experiments

In all experiments (unless specified elsewhere), the context size (*c*) was set to 4 utterances, smoothing window size (*s*) was set to 3 and cut-off threshold was set to $\mu - \sigma/2$; where μ is the mean of the depth scores and σ is their standard deviation. The performance of each sub-task boundary predictor is reported in terms of P_k, standard F-measure (F-1) and concept-based F-measure (C. F-1). The number reported is the average performance over all of the dialogs in the test corpus.

For a sub-task boundary predictor that gains substantial improvement over the baseline predictor, the significance level of the improvement is also reported. To avoid making an assumption that the segmentation performance on each dialog has a normal distribution, the Wilcoxon signed-rank test was adopted instead of the Student's t-test. If the Student's t-test is used when the data does not have a normal distribution, the p-value can be misleading. The Wilcoxon test is a nonparametric test; hence, it makes fewer assumptions on the distribution of the data. More detail discussions of both statistical hypothesis tests can be found in many statistics textbook such as (Degroot, 1986).

It has been argued that, a nonparametric test is less powerful than a parametric test that assumes a normal distribution (e.g. the Student's t-test), namely a nonparametric test is less likely to produces a small p-value. Therefore, it is more difficult to find the improvement that is statistically significant when a nonparametric test is used. In the preliminary experiments, I found that the p-values obtained from the Wilcoxon test were usually higher than the p-values obtained from the Student's t-test. A less powerful test, the Wilcoxon test, was chosen to ensure that the improvement achieved by the proposed approach is statistically significant. A Perl script provided by Institute of Phonetic Sciences, Amsterdam[7] was used to perform the Wilcoxon signed-rank test.

6.1.6.1 TextTiling baselines

Two baseline boundary predictors were created from the conventional TextTiling algorithm. The first predictor, B1 (B stands for baseline predictor), uses all of the words in the transcription as features. The second predictor, B2, excludes stop words specified by a hand-crafted list. The stop word list was taken from the list that was developed in the

[7] http://www.fon.hum.uva.nl/rob/SignedRank/SRTest.pl

Snowball project[8]. This stop word list contains 174 words and includes pronouns, prepositions and auxiliary verbs. The baseline performances are shown in Table 6.7.

Name	Stop word treatment	Air Travel			Map Task		
		P_k	F-1	C. F-1	P_k	F-1	C. F-1
B1	Include all words	0.384	0.427	0.654	0.412	0.239	0.396
B2	Exclude hand-crafted stop words	0.387	0.383	0.621	0.395	0.269	0.426

Table 6.7: TextTiling baseline performances

Summary

When both baseline predictors are applied in two different domains, the results reveals that a hand-crafted stop word list is not optimal for every domain. In the air travel domain, removing hand-crafted stop words actually decreases segmentation performance; however, the same stop word list does improve segmentation performance in the map reading domain.

6.1.6.2 Data-driven stop word treatments

T3 and T4 (T stands for TextTiling) are sub-task boundary predictors that use the data-driven approaches discussed in Section 6.1.1 to handle stop words. T3 excludes stop words from a set of features similar to B2 but uses a dialog-specific stop word list instead of a hand-crafted stop word list. The threshold for selecting dialog-specific stop words was set to $\mu + 2*\sigma$ in all of the experiments; where μ is the mean of the regularity counts of all the words in a given dialog and σ is their standard deviation. The average number of dialog-specific stop words is 8.8 words per dialog in the air travel planning domain and 9.5 words per dialog in the map reading domain. Regularity counts discover common words that are specific to spoken dialog domains, but are not listed in the hand-crafted stop word list, such as “okay” and “yeah”. However, some function words that occur only a few times are not included in the data-driven stop word list. The second predictor, T4, does not remove any word from context windows but weighs each word with a regularity weight computed by Equation (6.1). The performances of both sub-task boundary predictors are shown in Table 6.8.

[8] http://search.cpan.org/~creamyg/Lingua-StopWords-0.08/lib/Lingua/StopWords.pm.

Name	Stop word treatment	Air Travel			Map Task		
		P_k	F-1	C. F-1	P_k	F-1	C. F-1
T3	Exclude dialog-specific stop words	0.372	0.455	0.664	0.413	0.251	0.381
T4	Use regularity weights	0.384	0.420	0.641	0.428	0.231	0.379

Table 6.8: Results of data-driven stop word treatments

T3, which uses a dialog-specific stop word list, outperforms both baseline boundary predictors (B1 and B2) in the air travel domain. However, its performance is worse than both baseline predictors in the map reading domain. In the map reading domain, many content words also occur regularly throughout a dialog since a dialog composes of a series of the same type of sub-task, a **draw_a_segment** sub-task. Therefore, it is quite difficult to distinguish between content words and stop words. With the current cut-off threshold, some concept words such as “right” are removed while some common words such as “do” remains. Some concept words occur quite regularly but as different concept types; for example, some landmarks are **EndLocation** in one sub-task but are **StartLocation** in another sub-task. In another case, “right” can be both **Direction** and a non-concept word for acknowledgement. Therefore, information from concept annotation may help improve the performance of the predictor that identifies stop words from word distribution.

In T4, words that occur regularly are not removed unlike in T3, but are assigned only small weights. T4 performs slightly worse than T3 in both domains. Weighted common words still have some contribution to a cosine similarity score since there are many of them.

<u>Summary</u>

Data-driven stop word treatments can discover the stop words that are specific to the interested genre and domain. However, the efficiency of a data-driven stop word treatment is also depended on word distribution characteristic. In map reading domain, where the distributions of content words are quite similar to those of stop words, the data-driven approaches are less efficient.

6.1.6.3 Word token representation

The segmentation results of the predictors that use information from concept annotation, manually created, are shown in Table 6.9. TC1, TC2, TC3 and TC4 use the same stop word treatment as their counterparts, B1, B2, T3 and T4 respectively, but also incorporate information from concept annotation in word token representations. C in the predictor’s name stands for concept information.

Name	Stop word treatment	Air Travel			Map Task		
		P_k	**F-1**	**C. F-1**	**P_k**	**F-1**	**C. F-1**
TC1	Include all words	0.395	0.425	0.661	0.403	0.268	0.415
TC2	Exclude hand-crafted stop words	0.360	0.385	0.634	0.398	0.299	0.440
TC3	Exclude dialog-specific stop words	0.353	**0.461**	**0.695**	0.397	0.254	0.425
TC4	Use regularity weights	0.388	0.426	0.671	0.390	0.282	0.443

Table 6.9: Segmentation performances when incorporating concept information

In both domains, the boundary predictors that use information from concept annotation (TC1, TC2, TC3 and TC4) perform better than their counterparts (B1, B2, T3 and T4) that do not use the information, especially when evaluated by concept-based F-1. TC3 and TC4, which use a data-driven approach to define stop words from word distribution, can benefit more from concept information than TC1 and TC2, which use a traditional stop word treatment. Information from concept annotation provides a richer representation that helps distinguish between different concept types and also between different word senses. Co-occurrences of the same word string that actually belong to dissimilar concepts or dissimilar worse senses no longer affect the similarity score. In addition, this better representation makes the distributions of content words more distinguishable from the distributions of stop words; therefore, the boundary predictors that use data-driven stop word treatments, which rely on word distribution characteristic, can achieve more performance improvement.

In the air travel domain, both TC3 and TC4 achieve better segmentation results than TC1 and TC2. When compared to the baselines, TC3, which uses information from concept annotation and a dialog-specific stop word list, performs significantly better than B2, which uses a hand-crafted stop word list, both in terms of exact F-1 (p-value = 0.002) and concept-based F-1 (p-value = 0.028). The result which is significantly better than the baseline at a significance level of 0.05 (an α-level of 5%) is highlighted in bold[9]. In the map reading domain, the performances of TC3 and TC4 are about the same as the performances of TC1 and TC2.

[9] Since there are two baseline boundary predictors in the experiment, a significance test is conducted for each baseline separately. When a statistically significant improvement over one of the baseline is found, the p-value and the significance level of that particular test are reported. The claim of significant improvement is made specifically for that particular baseline to avoid a multiple testing problem.

Summary

Information from concept annotation improves dialog segmentation results in both domains as it provides better word token representation that can distinguish between different concept types and also between different word senses. This better representation also makes the data-driven stop word treatments more efficient especially in the map reading domain. While a performance of the boundary predictor that uses a hand-crafted stop word list depended on compatibility between the selected stop word list and the domain, the performance of the boundary predictor that uses a data-driven stop word treatment (a dialog-specific stop word list or regularity weights) is not since the stop words are determined directly from word distribution in each data set.

6.1.6.4 Distance weight

The segmentation results obtained when distance weights are applied to sub-task boundary predictors are shown in Table 6.10. D in the predictor's name stands for a distance weight. These boundary predictors do not use information from concept annotation and should be compared to B1, B2, T3 and T4. Distance weights improve the results, as measured by concept-based F-1, for most of the cases. The improvement comes from higher recall. Distance weights demote irrelevant similarity from far away contexts; therefore, help the TextTiling algorithm discovers more boundaries.

Name	Stop word treatment	Air Travel			Map Task		
		P_k	F-1	C. F-1	P_k	F-1	C. F-1
T1D	Include all words	0.399	0.418	0.636	0.412	0.241	0.412
T2D	Exclude hand-crafted stop words	0.384	0.397	0.659	0.397	0.273	0.435
T3D	Exclude dialog-specific stop words	0.395	0.435	0.672	0.409	0.239	0.400
T4D	Use regularity weights	0.374	**0.438**	**0.696**	0.414	0.226	0.404

Table 6.10: Segmentation performances when distance weights are applied to the predictors that do not use concept information.

T4D which uses regularity weights together with distance weights obtains significant improvement over the baseline that uses a hand-crafted stop word list (B2) both in terms of exact match or standard F-measure (p-value = 0.026) and concept-based F-1 (p-value = 0.008). However, in the map reading domain, the boundary predictors that use data-driven stop word treatments are still not better than the B2 baseline. T3D and T4D have the same problem as T3 and T4 that is without information form concept annotation the distributions of content words are quite similar to those of stop words which make the data-driven approaches less efficient.

Name	Stop word treatment	Air Travel			Map Task		
		P_k	F-1	C. F-1	P_k	F-1	C. F-1
TC1D	Include all words	0.405	0.387	0.628	0.399	0.260	0.423
TC2D	Exclude hand-crafted stop words	0.362	0.399	0.685	0.380	0.290	0.456
TC3D	Exclude dialog-specific stop words	0.357	**0.482**	**0.720**	0.391	0.247	0.418
TC4D	Use regularity weights	0.377	0.434	**0.694**	0.371	0.278	**0.470**

Table 6.11: Segmentation performances when applied distance weights to the predictors that used concept information.

Table 6.11 presents the segmentation results obtained when distance weights are applied to the predictors that also utilize information from concept annotation. Distance weights improve the segmentation performance when compared to the results in Table 6.9, where the distance weights are not applied, as measured by concept-based F-1 for most of the cases. In the air travel domain, the predictors that use data-driven stop word treatments (TC3D and TC4D) achieve a significant improvement over the baseline that uses a hand-crafted stop word list (B2), in terms of concept-based F-measure (p-value = 0.001 for TC3D and p-value = 0.004 for TC4D). TC3D also gains a significant improvement over the baseline (B2) in terms of exact F-1 (p-value = 0.002). In the map reading domain TC4D achieves a significant improvement over the baseline predictor does not remove any stop word (B1) in terms of concept-based F-measure (p-value = 0.007). The segmentation result obtained from TC4D is also slightly better than the result obtained from TC2D which uses a hand-crafted stop word list.

Summary

The boundary predictors that utilize distance weights achieve better segmentation results, in terms of concept-based F-1, than the predictors that do not use distance weights. The performance gain can be achieved by both the predictors that use information from concept annotation and the ones that do not use it. When distance weights are used together with a data-driven stop word treatment, a significant improvement over the baseline can be achieved in both domains.

6.1.6.5 Context size

To examine the effect of the amount of context on segmentation performance, in this experiment, the context window size used in cosine similarity calculation was varied from 4 utterances to 16 utterances. To also observe the effect of distance weights on segmentation performance when the size of the context window increases, two boundary predictors, TC4 and TC4D, were used. The former uses regularity weights and information from concept annotation while the latter uses additional distance weights. Both predictors achieved good performance in both domains as shown in the previous experiments. The segmentation results in the air travel domain are shown in Table 6.12, Figure 6.4 and Figure 6.5.

Context size	w/o distance weight (TC4)				with distance weight (TC4D)			
	P_k	C. Prec	C. Recall	C. F-1	P_k	C. Prec	C. Recall	C. F-1
4	0.388	0.685	0.702	0.671	0.377	0.689	0.747	0.694
6	0.431	0.675	0.614	0.620	0.392	0.669	0.693	0.657
8	0.394	0.740	0.650	0.674	0.403	0.698	0.677	0.662
10	0.410	0.711	0.581	0.624	0.424	0.693	0.608	0.628
12	0.408	0.745	0.581	0.632	0.415	0.719	0.599	0.634
14	0.410	0.766	0.610	0.656	0.410	0.738	0.628	0.657
16	0.409	0.777	0.609	0.661	0.425	0.737	0.616	0.648

Table 6.12: The performances of sub-task boundary predictors when the context size is varied in the air travel domain.

When a larger context window is used, a context vector changes more gradually from one sliding window to another sliding window which makes the similarity score curve smoother. Both TC4 and TC4D predict less number of boundaries as the window size increases. As a result, precision tends to increase while recall tends to decrease as shown in Figure 6.4 and Figure 6.5. Similar observation is also found in the map reading domain. The trade-off between precision and recall determines the overall performance of the predictors.

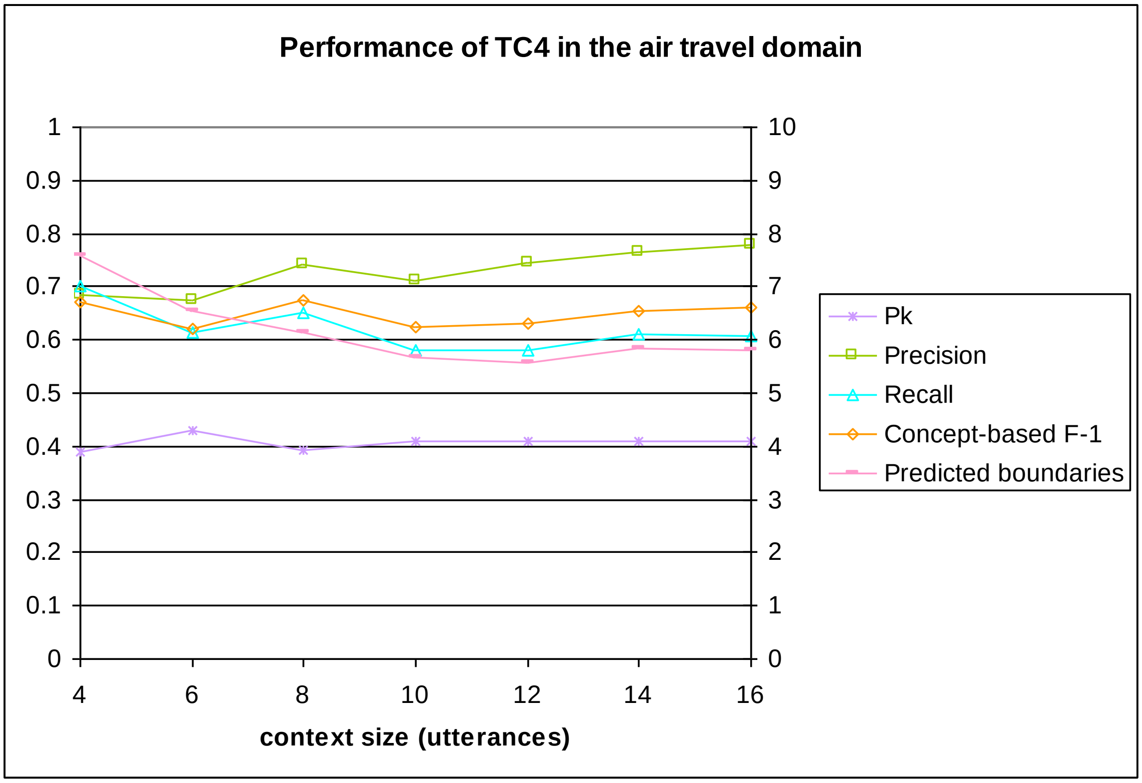


Figure 6.4: The effect of the context size on the TC4 predictor in the air travel domain.

In the air travel domain, when the context size increases, the overall performance of both boundary predictors decreases slightly both in terms of P_k and concept-based F-1. The performance is not monotonic decreasing; however, the performance is optimal when the context window size is set to 4 utterances. The loss in recall is greater than the gain in precision when the context size is large. When the context size increases, both predictors miss more boundaries between small sub-tasks (for example, between a **query_car_info** sub-task and a **query_hotel_info** sub-task). If the context windows are larger than the sub-tasks, they are more likely to contain irrelevant context and irrelevant similarity from other sub-tasks. This irrelevant similarity may prevent the TextTiling algorithm from detecting the boundary between two small sub-tasks since there is no significant drop in cosine similarity scores.

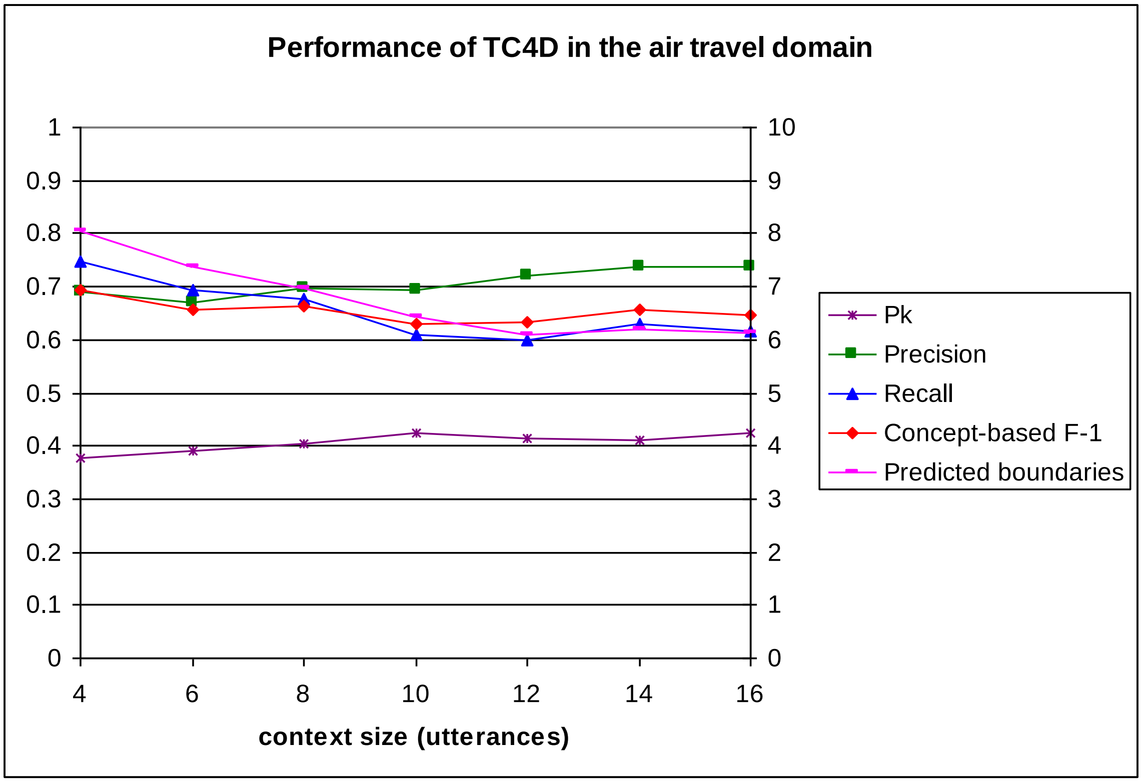


Figure 6.5: The effect of the context size on the TC4D predictor in the air travel domain.

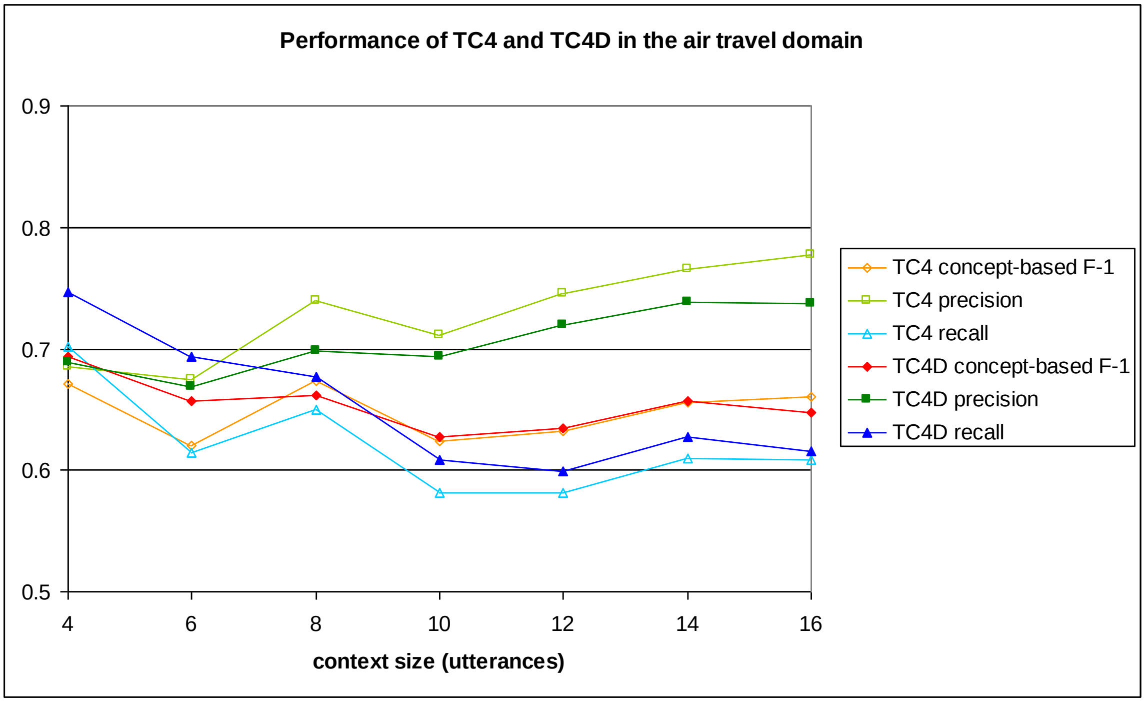


Figure 6.6: The effect of distance weights when the context size is varied in the air travel domain

The performances of TC4 and TC4D predictors in terms of concept-based F-1 are compared in Figure 6.6. The graphs for TC4 are in lighter colors while the graphs for TC4D are in darker colors. TC4D, which uses distance weights, achieves better performance from higher recall when the context window is small. However, when the context window is larger, the recall and concept-based F-1 of both predictors drop to about the same values.

An error analysis reveals that longer dialogs benefit more from a larger context window. A long dialog usually has longer sub-tasks (from lengthy negotiation) rather than more sub-tasks. With a larger context window, the TextTiling algorithm is able to detect similar words within to the same sub-task that are quite far apart and hence reduces the number false alarms without missing more boundaries.

To investigate this observation more closely, I separated the dialogs into two groups according to their length. The first group contains the dialogs that are shorter than the average dialog length (55.63 utterances) while the second group contains the dialogs that are longer than the average. When the size to of the context window increases, the segmentation performance decreases in the first group but increases in the second group as shown in Table 6.13 and Figure 6.7. The performances of short dialogs are illustrated with lighter color graphs. Similar results are obtained for both TC4 and TC4D. The segmentation performance for the set of long dialogs is optimal when the context window size is set to 16 utterances for both TC4 and TC4D. For TC4, the performance when using a large context window (context size = 16) is significantly better than when using a small context window (context = 4) in terms of P_k (p-value = 0.008).

Context size	w/o distance weight (TC4)				with distance weight (TC4D)			
	short dialogs		long dialogs		short dialogs		long dialogs	
	P_k	C. F-1	P_k	C. F-1	P_k	C. F-1	P_k	C. F-1
4	0.380	0.698	0.404	0.616	0.359	0.723	0.411	0.637
6	0.474	0.617	0.345	0.627	0.398	0.669	0.381	0.633
8	0.418	0.679	0.347	0.665	0.423	0.659	0.363	0.668
10	0.442	0.626	0.347	0.619	0.440	0.623	0.391	0.638
12	0.452	0.614	0.321	0.666	0.433	0.630	0.380	0.642
14	0.454	0.646	0.322	0.675	0.434	0.648	0.362	0.674
16	0.465	0.650	0.299	0.684	0.462	0.628	0.350	0.689

Table 6.13: Performance comparison between short dialogs and long dialogs in the air travel domain

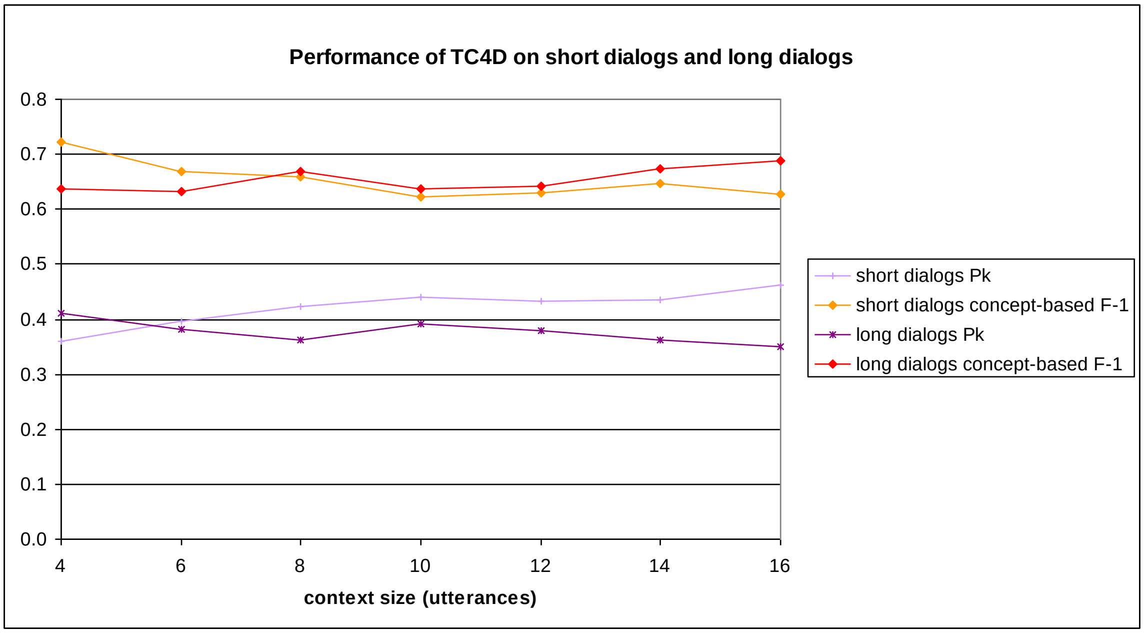


Figure 6.7: The effect of the context size on short dialogs and long dialogs in the air travel domain

Segmentation performances of TC4 and TC4D predictors in the map reading domain are shown in Table 6.14 and Figure 6.8. As the context size increases, both TC4 and TC4D predict fewer boundaries. Therefore, precision increases at the expense of recall, similar to the results obtained in the air travel domain. The boundary predictor that does not use distance weights (TC4) achieves a smaller gain in precision than the reduction in recall; hence, segmentation performance is slightly lower as the context size increases. The predictor that uses distance weights (TC4D), on the other hand, achieves higher gain in precision that can compensate for the reduction in recall; therefore, its segmentation performance does not change much. In this domain, TC4D can still recover more boundaries than TC4 even when a larger context window is used unlike in the air travel domain. Another difference is that regardless of the amount of the context used, there is only small difference in segmentation performance between short dialogs and long dialogs.

Context size	**w/o distance weight (TC4)**				**with distance weight (TC4D)**			
	P_k	**C. Prec**	**C. Recall**	**C. F-1**	**P_k**	**C. Prec**	**C. Recall**	**C. F-1**
4	0.390	0.490	0.409	0.443	0.371	0.499	0.450	0.470
6	0.372	0.523	0.386	0.440	0.371	0.538	0.427	0.472
8	0.385	0.532	0.367	0.430	0.371	0.544	0.410	0.464
10	0.382	0.532	0.362	0.425	0.360	0.558	0.426	0.478
12	0.371	0.555	0.362	0.432	0.379	0.539	0.395	0.450
14	0.369	0.540	0.375	0.438	0.368	0.573	0.409	0.474
16	0.391	0.539	0.349	0.421	0.363	0.578	0.397	0.467

Table 6.14: The performances of both predictors when the context size is varied in the map reading domain.

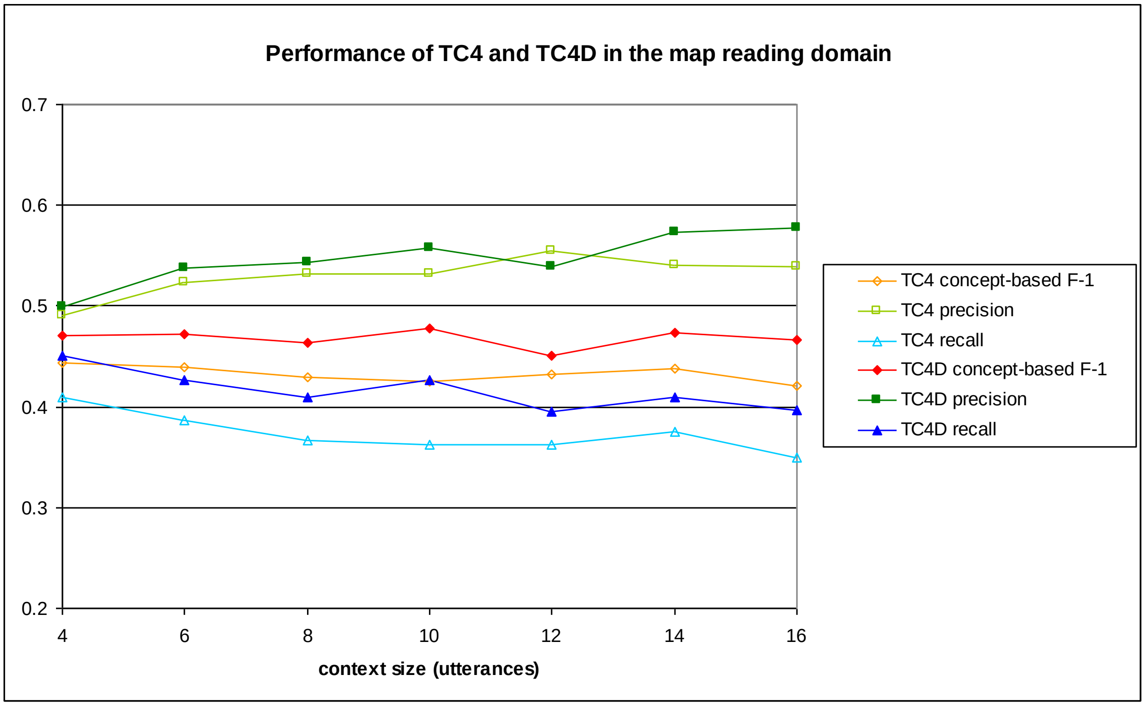


Figure 6.8: The effect of distance weights when the context size is varied in the map reading domain

There are two types of sub-tasks in the map reading domain: **draw_a_segment** and **grounding**. **Grounding** is a finer-grained sub-task embedded inside a **draw_a_segment** sub-task. The discussion on task structure decomposition in the map reading domain is presented in Section 3.4. When the context size increases, both TC4 and TC4D miss more boundaries between finer-grained sub-tasks (**grounding** sub-tasks) than the boundaries between coarser-grained sub-tasks (**draw_a_segment** sub-tasks). To evaluate segmentation performance on coarse-grained sub-tasks, only the boundaries of **draw_a_segment** sub-tasks are considered; the boundaries of **grounding** sub-tasks are excluded from the reference boundaries. The performances of TC4 and TC4D boundary predictors on coarse-grained sub-tasks are shown in Table 6.15.

Context size	w/o distance weight (TC4)				with distance weight (TC4D)			
	P_k	C. Prec	C. Recall	C. F-1	P_k	C. Prec	C. Recall	C. F-1
4	0.439	0.365	0.529	0.427	0.422	0.375	0.589	0.450
6	0.409	0.392	0.495	0.432	0.406	0.412	0.568	0.470
8	0.395	0.399	0.475	0.430	0.417	0.425	0.551	0.472
10	0.376	0.419	0.485	0.444	0.391	0.430	0.567	0.482
12	0.372	0.439	0.489	0.455	0.384	0.426	0.529	0.465
14	0.385	0.419	0.498	0.450	0.378	0.453	0.557	0.492
16	0.371	0.430	0.477	0.447	0.353	0.473	0.558	0.504

Table 6.15: Segmentation performances on coarse-grained sub-tasks in the map reading domain

An average segment length of coarse-grained sub-tasks is 13.8 utterances while an average segment length of all of sub-tasks is 7.4 utterances. When the context size increases, both boundary predictors have slightly better performance on coarse-grained sub-tasks due to reduction in false alarms. Between the two predictors, TC4D gains more improvement in precision than TC4 and also has better overall performance. When a larger context window is used, TC4D misses only a few more boundaries but reduces the number of false alarms by about one third.

Summary

A sub-task boundary predictor that uses a large context window can achieve a better result in specific circumstances. In the air travel planning domain, a large context window improves segmentation performance in long dialogs. In the map reading domain, a large context window is more suitable for identifying coarser-grained sub-task boundaries. In the map reading domain, distance weights help recover more boundaries regardless of the context window size used; however, in the air travel domain, there is no performance gain from distance weights when the context window size is large.

Using an appropriate amount of context is crucial when segmenting a discourse into fine-grained segments such as sub-tasks. Even though a boundary predictor that uses a large context window can achieve better performance in some specific cases, a better overall performance can be achieved with a small context window. A small context window is more sensitive to small changes in the context and thus suitable for identifying fine-grained segments particularly when there is high variation in the segment length as in the air travel planning domain.

6.1.6.6 Smoothing algorithm

In this experiment, two similarity score smoothing algorithms were examined: rectangular smoothing and triangular smoothing. Both smoothing algorithms are described in Section 6.1.2.2. Rectangular smoothing was used as a default smoothing technique in all previous experiments. A triangular smoothing technique is applied in T4DTr and TC4DTr. The former uses only transcribed words while the latter also incorporates information from concept annotation. Both predictors utilize regularity weighs and distance weights similar to T4D and TC4D but use a triangular smoothing scheme instead of a rectangular smoothing scheme (Tr stands for triangular smoothing). The results of T4D and TC4D presented in Table 6.16 are similar to the ones presented in Table 6.10 and Table 6.11 respectively.

Name	**Smoothing technique**	**Air Travel**			**Map Task**		
		P_k	**F-1**	**C. F-1**	**P_k**	**F-1**	**C. F-1**
T4D	Rectangular smoothing	0.374	**0.438**	**0.696**	0.414	0.226	0.404
T4DTr	Triangular smoothing	0.376	**0.445**	**0.702**	0.424	0.230	0.396
TC4D	Rectangular smoothing	0.377	0.434	**0.694**	0.371	0.278	**0.470**
TC4DTr	Triangular smoothing	0.371	**0.457**	**0.712**	0.384	**0.292**	**0.464**

Table 6.16: Segmentation results when used different smoothing algorithms

The boundary predictors that utilize a triangular smoothing scheme achieve slightly better exact F-1 than the predictors that use a rectangular smoothing scheme (but not significantly better). However, P_k and concept-based F-1 are not always improved. P_k and concept-based F-1 already award some credit to near missed boundaries; therefore, a more precise boundary prediction may not always improve their values.

Since the boundary predictors that use a triangular smoothing algorithm can achieve about the same level of concept-based F-1 as the predictors that use a rectangular smoothing algorithm, they also gain a significant improvement over the baseline, similar to their counterparts. In the air travel domain, both T4DTr and TC4DTr are significantly better than the baseline predictor that uses a hand-crafted stop word list (B2). The p-values are 0.003 and 0.002 respectively. In the map reading domain, TC4DTr is significantly better than the baseline that includes all words in the transcript as features (B1); p-value is equal to 0.026. The result which is significantly better than the baseline at a significance level of 0.05 (an α-level of 5%) is highlighted in bold.

In terms of exact F-1, the boundary predictors that use a triangular smoothing algorithm achieve more improvement when compared to the baselines than the predictors that use a rectangular smoothing algorithm. In the air travel domain, both T4DTr and TC4DTr achieve a significant improvement over the baseline B2; the p-value is 0.015 and 0.003 respectively. In the map reading domain, the p-value of TC4DTr is 0.050.

Summary

A triangular smoothing algorithm produces a more precise boundary prediction than a rectangular smoothing algorithm as shown by the improvement in exact F-1.

6.1.6.7 Cut-off threshold

It has been observed during the error analysis of segmentation results that sometimes the TextTiling algorithm found a drop in similarity scores (a valley) at the location where the boundary should be; however, its depth score was not high enough for the algorithm to predict it as a boundary. If the cut-off threshold is lower, these missing boundaries could be discovered; nevertheless, this might come with the cost of higher false alarms.

In this experiment the effect of a cut-off threshold on segmentation performance is investigated. Four cut-off thresholds were examined. *Threshold-1* is the threshold that was used in all previous experiments. The cut-off threshold for selecting a boundary is set to $\mu - \sigma/2$; where μ is the mean of the depth scores and σ is their standard deviation. *Threshold-2* has a lower value than *threshold-1*; the cut-off threshold is set to $\mu – \sigma$. For *all*, there is no cut-off threshold; all of the valleys in the similarity score plot are outputted as boundaries. For *oracle*, the top-*n* candidate boundaries that have the highest depth scores are outputted; *n* is the number of reference boundaries in each dialog. The sub-task boundary predictor used in this experiment is the predictor that uses information from concept annotation, regularity weights and distance weights together with a triangular smoothing algorithm (TC4DTr). This boundary predictor is discussed in the previous section. Table 6.17 and Figure 6.9 present the segmentation results of TC4DTr in the air travel domain when different cut-off thresholds are used. The results in the map reading domain are presented in Table 6.18 and Figure 6.10. The result of *threshold-1* is the same as the result of TC4DTr shown in Table 6.16.

Threshold	#Predicted boundaries	#Predicted/ #reference ratio	P_k	Concept-based		
				Precision	Recall	F-1
threshold-1	8.125	1.266	0.371	0.693	0.771	0.712
threshold-2	9.292	1.448	0.393	0.630	0.803	0.692
all	11.167	1.740	0.423	0.567	0.840	0.666
oracle	6.417	1.000	0.362	0.729	0.685	0.704

Table 6.17: Segmentation performance when different cut-off thresholds are used in the air travel domain

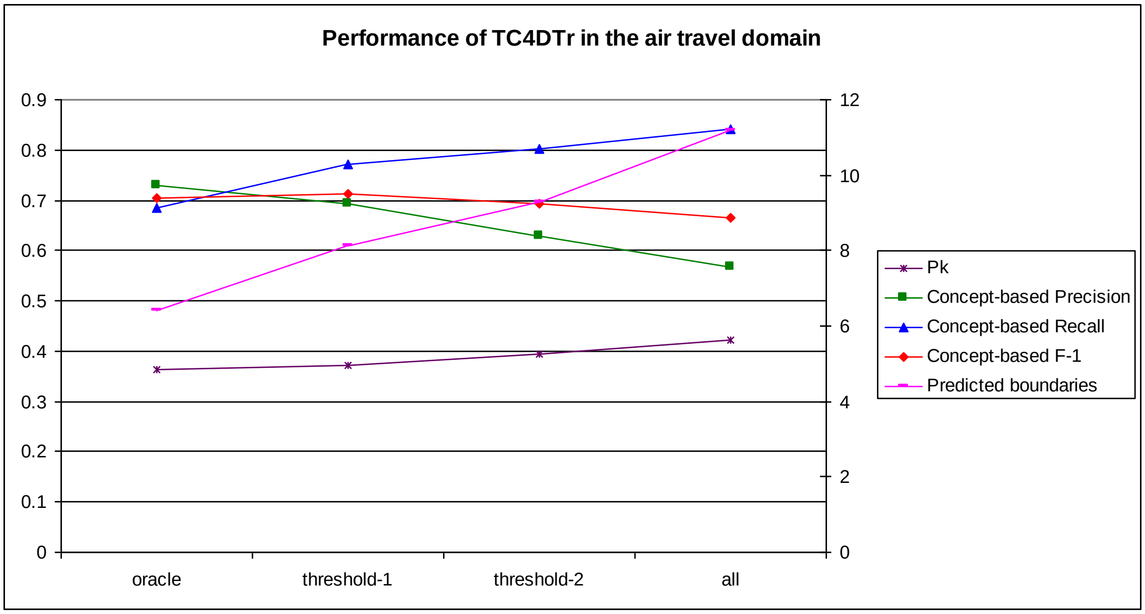


Figure 6.9: The effect of the cut-off threshold in the air travel domain

Threshold	#Predicted boundaries	#Predicted/ #reference ratio	P_k	Concept-based		
				Precision	Recall	F-1
threshold-1	15.550	0.861	0.384	0.506	0.433	0.464
threshold-2	19.650	1.089	0.375	0.480	0.513	0.492
all	23.150	1.283	0.372	0.461	0.580	0.511
oracle	18.050	1.000	0.366	0.498	0.493	0.496

Table 6.18: Segmentation performance when different cut-off thresholds are used in the map reading domain

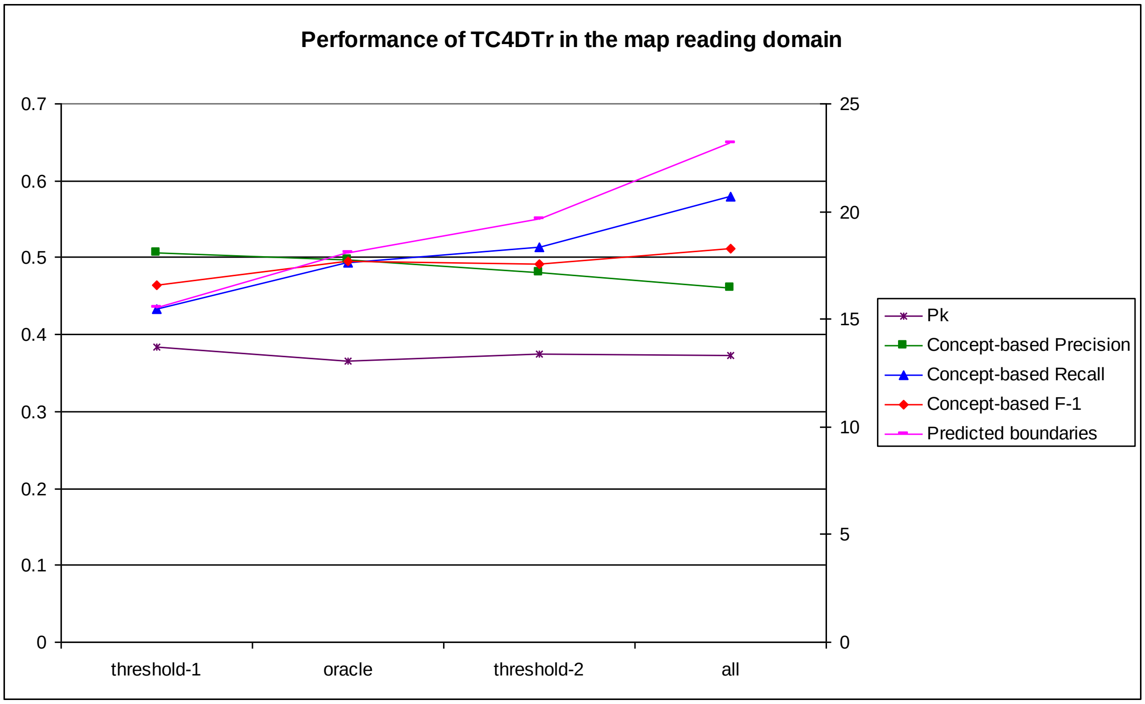


Figure 6.10: The effect of the cut-off threshold in the map reading domain

In Figure 6.9 and Figure 6.10, the threshold that yields fewer predicted boundaries is presented first. As expected when more boundaries are predicted recall is higher while precision is lower. Therefore, the overall performance is a trade-off between the gain in recall and the loss in precision. When the boundary predictor outputs more boundaries, the overall performance decreases in the air travel domain but increases in the map reading domain. The third column of Table 6.17 and Table 6.18 show the ratio between the number of predicted boundaries and the number of reference boundaries in the air travel domain and the map reading domain respectively. This ratio is used to compare the amount of boundaries outputted by a sub-task boundary predictor when the sets of input dialogs are not the same. When the same thresholding method is used, The predictor outputs less number of boundaries in the map reading domain as indicates by the lower predicted boundary ratio. A significant drop in lexical similarity, which indicates a potential segment boundary, occurs less frequent because the consecutive sub-tasks in the map reading domain are more similar given that each dialog composes of a series of the same type of sub-task. Therefore, a lower threshold is more suitable is the map reading domain.

One interesting observation is that the *oracle* threshold does not provide optimal performance in terms of concept-based F-1, but the threshold that outputs slightly more boundaries than the reference boundaries does. Since the boundary predictor can generate some false alarms, allowing the predictor to output slightly more boundaries than the number of reference boundaries can achieve higher gain in recall than reduction in precision assuming that the candidate boundaries that have high depth scores are likely to be true boundaries.

Summary

An appropriate cut-off threshold for predicting a sub-task boundary from the depth score is depended on the characteristic of the domain. If consecutive sub-tasks in a dialog are quite similar, a lower threshold is required as in the map reading domain. Similarly, a domain which has fine-grained sub-tasks may also require a low cut-off threshold.

6.1.6.8 Error analysis

A dialog segmentation result obtained from a TC4DTr sub-task boundary predictor which achieve good segmentation results in both the air travel domain and the map reading domain was analyzed. This predictor uses information from concept annotation, regularity weights and distance weights together with a triangular smoothing algorithm and is discussed in Section 6.1.6.6. There are two types of segmentation errors: missing boundaries and false alarms.

Left segment	Right segment	Missed boundaries	
		Count	%
query_flight_info	query_flight_info	14	53.85
query_flight_info	query_flights_fare	7	31.82
query_flights_fare	query_car_info	5	55.56
query_car_info	query_hotel_info	4	40.00

Table 6.19: The most frequent missing boundaries in the air travel domain

Table 6.19 presents the most frequent missing boundaries in the air travel domain. The first and the second column show the left sub-task and the right sub-task of the missing boundary respectively. There are two types of boundaries that are difficult to identify by the TextTiling algorithm. The first one is a boundary between two consecutive sub-tasks of the same type such as a boundary between two **query_flight_info** sub-tasks that represent a departing flight and a return flight. Dialog segments that belong to the same sub-task type are more similar than dialog segments that belong to different sub-task types. Therefore, based on the lexical coherence assumption, it is more difficult to identify the boundary between two instances of the same sub-task type.

The second problem is the boundaries of a small sub-task that is only 2-3 utterances long such as a **query_flights_fare** sub-task. It is difficult to accurately identify the boundaries of fine-grained segments since useful context is limited. Moreover, the difficulty also comes from the limitation of the TextTiling algorithm. In order to identify both boundaries of a small sub-task, there must be two significant drops in a cosine similarity score plot that are only a couple of utterances apart. The TextTiling algorithm may be able to identify one boundary of a small sub-task but usually fail to detect another boundary since it is unlikely to have another significant drop in lexical cohesion that is very close to the first one. A boundary between a **query_car_info** sub-task and a **query_hotel_info** sub-task is sometimes difficult to identify since some instances of a **query_car_info** sub-task and a **query_hotel_info** sub-task are quite short. Moreover, they sometimes contain similar concepts and keywords, such as **[Fare]:**dollar and “rate”.

Boundary type	**Missed boundaries**	
	Count	**%**
draw_a_segment	89	44.28%
grounding (embedded)	117	73.13%

Table 6.20: Missing boundaries in the map reading domain

Statistic of missing boundaries in the map reading domain is shown in Table 6.20. Since there are only two types of sub-tasks in this domain and a **grounding** sub-task is embedded inside a **draw_a_segment** sub-task, the boundaries can be categorized into two types: a boundary between two **draw_a_segment** sub-tasks and a boundary of a **grounding** sub-task inside a **draw_a_segment** sub-task. The type of a boundary is indicated in the first column of Table 6.20. Boundaries of **grounding** sub-tasks are more problematic since the sub-tasks are quite short; some of them are only 2-3 utterances long. This problem is similar to the one occurs to the boundaries of **query_flights_fare** sub-tasks in the air travel domain discussed above. Since most of the boundaries in the map reading domain belong to those two problematic cases, the segmentation performance in the map reading domain is lower than the performance in the air travel domain.

A false alarm sometimes occurs in a long segment since a long discussion usually has a slight shift in topic that might be detected by the TextTiling algorithm. In the air travel domain some false alarms are correlated with sub-structures within a long sub-task. For example, some false alarms occur between a dialog segment that discusses a departure city and a dialog segment that discusses departure date and time. Some false alarms occur between a dialog segment that specifies all of the criteria for retrieving flight information and a dialog segment that discusses the results retrieved from the database.

6.1.7 HMM-based segmentation experiments

In all experiments, unless specified elsewhere, the parameters that were used to train a hidden Markov model are given in the following table.

Parameter	**Value**
Bisecting cluster re-assignment threshold (threshold-1)	5%
Number of bisecting runs (B)	10
Minimum cluster size (T)	5
Smoothed count for emission probability estimation (δ_1)	1
Smoothed count for transition probability estimation (δ_2)	1
Maximum HMM parameter re-estimating iterations	15
State label re-assignment threshold for Viterbi decoding and HMM parameter re-estimation iterations	1%

Table 6.21: Default values of the parameters in HMM training

The number of HMM states is another crucial parameter in HMM-based segmentation. In the implementation of the HMM-based approach employed in the experiments, only the maximum number of HMM states (*M*) can be specified. The actual number of HMM states (*m*) may be lower if the clustering algorithm combines small clusters into an etcetera state when it induces the initial HMM states from data. The value of *M*, the maximum number of HMM states, is treated as an independent variable in most experiments in order to investigate the effect of the number of HMM states on segmentation performance. For simplicity, when the number of HMM states is discussed in the following experiments, it refers to the maximum number of HMM states (*M*) not the actual number of HMM states (*m*) since *M* is the parameter that can be controlled in the experiments.

Since the bisecting *K*-means clustering algorithm contains a random process as the new centroids of split clusters are chosen randomly, the performance of a HMM-based segmentation is averaged over 10 runs.

6.1.7.1 Utterance-based segmentation

In this experiment, the initial states of the HMM-based sub-task boundary predictor are created by clustering similar utterances in the corpus together. A HMM is trained from dialog transcription that also contains concept annotation as it has been shown in the TextTiling experiment (Section 6.1.6.3) that information from concept annotation is useful for determining sub-task boundaries. In terms of a stop word treatment, the stop words that are specified in the hand-crafted list are removed. The effect of different stop word treatments on the segmentation performance is investigated in Section 6.1.7.4. The segmentation result of the HMM-based sub-task boundary predictor, UC2, is shown in Table 6.22 and Figure 6.11. The boundary predictor naming scheme is similar to the one used in TextTiling experiments where U stands for utterance-based HMM segmentation, C stands for concept information and the running number represents a stop word treatment.

Number of states (M)	**Predicted boundaries**	**Segment Length**	**P_k**	**Concept-based**		
				Precision	**Recall**	**F-1**
2	5.633	12.006	0.398	0.797	0.544	0.624
4	24.104	2.408	0.426	0.395	0.930	0.542
6	25.079	2.310	0.420	0.394	0.951	0.544
8	26.746	2.161	0.430	0.375	0.955	0.525
TextTiling	7.792	8.190	0.360	0.625	0.679	0.634

Table 6.22: Segmentation performance of an utterance-based HMM predictor (UC2) in the air travel domain

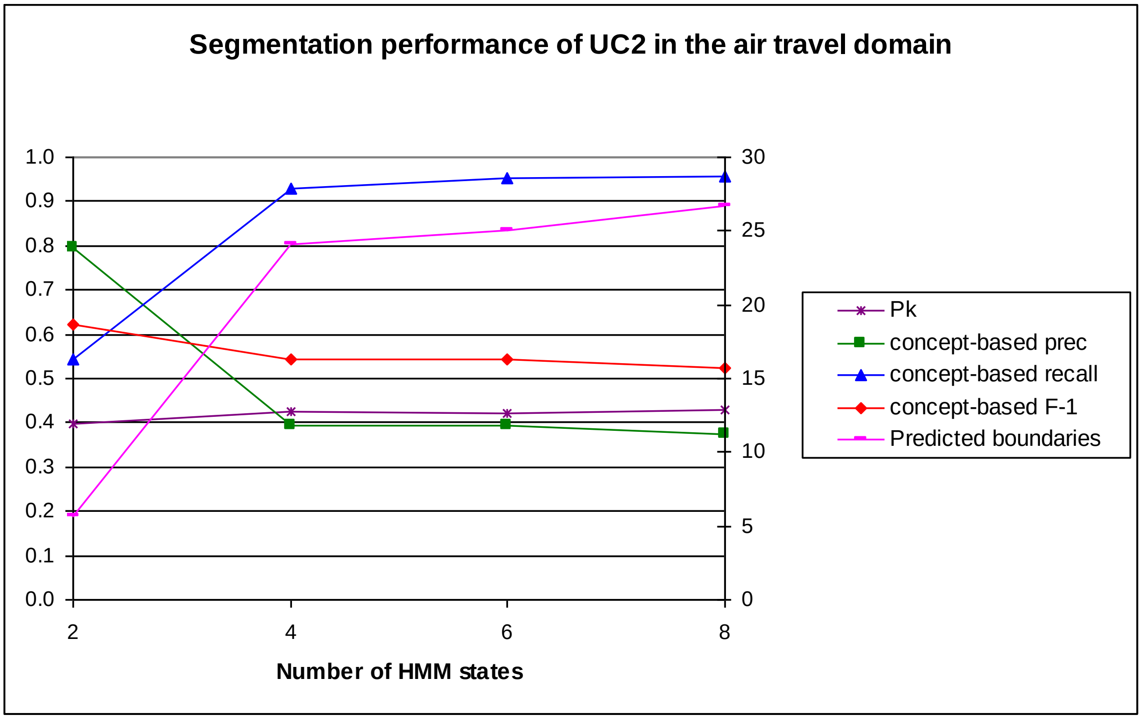


Figure 6.11: The effect of the number of HMM states on segmentation performance in the air travel domain

When more states are used, the HMM predictor tends to output more boundaries which increases the recall but at the same time deteriorates the precision. The overall performances are lower both in terms of P_k and concept-based F-1 as the loss in precision is greater than the gain in recall. The decision to limit the number of states to 8 came from a preliminary experiment, where the initial result indicated that segmentation performance did not change much when M is higher than 8 as measured by all evaluation metrics discussed in Section 6.1.4. The segmentation results obtained from the HMM-based boundary predictor are not as good as the result obtained from the TextTiling algorithm that uses the same features (TC2). The performance of the TextTiling algorithm is shown in the last row of Table 6.22. Please note that other boundary predictors that use different features can achieve better performance than TC2.

When M = 2, UC2 predicts much fewer boundaries than other cases. Most of the utterances from a **query_flight_info** sub-task are clustered into one state while the utterances from other sub-tasks are clustered into another state. Although, the precision is quite high, the predictor misses 85.8% of the boundaries between two **query_flight_info** sub-tasks. When M is higher ($M \geq 4$), UC2 predicts boundaries at every other utterance while the average true segment length is 10.3 utterances. Since the boundary is predicted between any two utterances such that their state labels are different, the number of predicted boundaries reflects the number of changes between HMM states as a sequence of utterances in a dialog is observed. The clusters induced by grouping similar utterances together are not homogeneous. In task-oriented dialogs, social phrases such as an acknowledgement and a yes/no response can occur in any sub-task. Therefore, these utterances, even though they belong to different sub-tasks, are more similar and are grouped together into the same state while utterances from the same sub-task may be clustered into different states. Moreover, many short utterances do not contain enough context to robustly determine the similarity. From those mentioned problems, the decoded state labels of the utterances that belong to the same sub-task may be different from one another and thus creates a lot of false alarms. When the number of HMM states is higher, it is more likely for the utterances from the same sub-task to be clustered into different states; therefore, the HMM-based predictor is more likely to output more boundaries.

The segmentation result from the map reading domain is shown in Table 6.23 and Figure 6.12. The performance of the TextTiling algorithm that uses the same features (TC2) is shown in the last row of the table. When more states are used, the number of predicted boundaries tends to increase but does not monotonically increase, as in the air travel domain. For all *M* values, the HMM-based predictor outputs much more boundaries than the average number of reference boundaries which is 18.1 boundaries per dialog; therefore, the number of false alarms is very high. When compared to the result obtained from the TextTiling algorithm that uses the same features (TC2), the HMM-based predictors have better recall but lower precision. When the number of states is equal to 8, the overall performance of the HMM-based predictor is about the same as the overall performance of the TextTiling algorithm. However, the HMM predicts too many boundaries, one boundary at every other utterance, to be considered useful.

Number of states (M)	Predicted boundaries	Segment Length	P_k	Concept-based		
				Precision	Recall	F-1
2	37.895	3.402	0.397	0.326	0.565	0.385
4	81.485	1.559	0.424	0.249	0.903	0.384
6	66.850	1.906	0.406	0.317	0.844	0.430
8	66.295	1.922	0.392	0.306	0.868	0.436
TextTiling	14.750	9.127	0.398	0.497	0.400	0.440

Table 6.23: Segmentation performance of an utterance-based HMM predictor (UC2) in the map reading domain

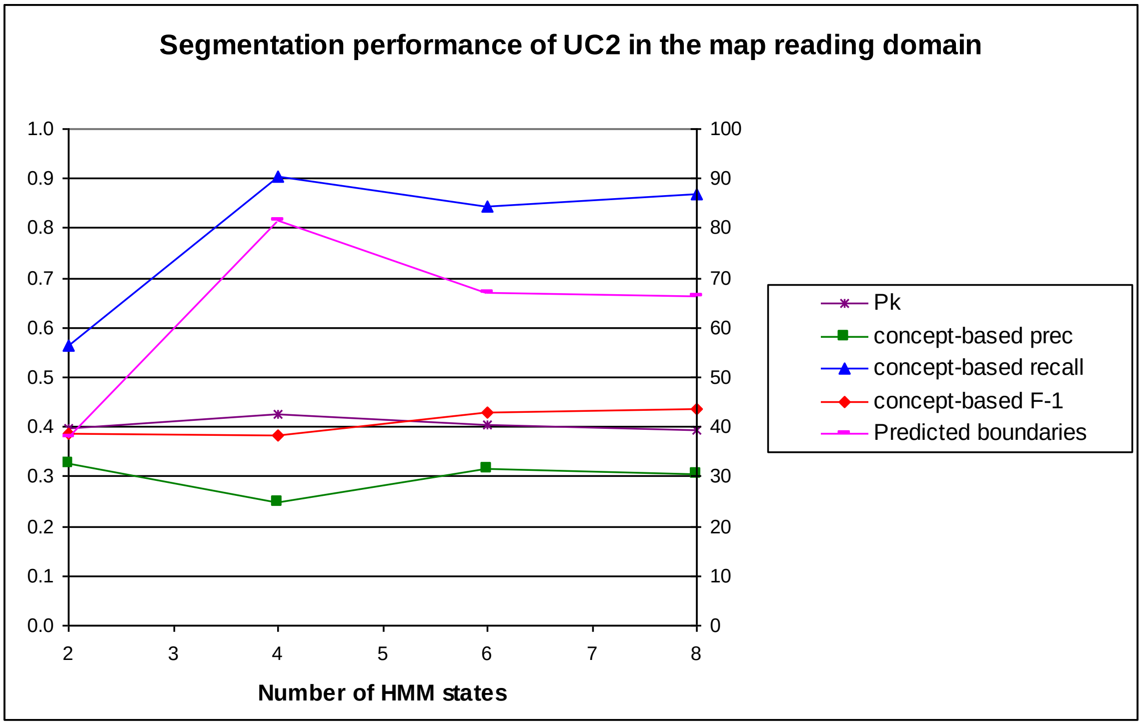


Figure 6.12: The effect of the number of HMM states on segmentation performance in the map reading domain

Summary

When HMM states are induced from utterances, the HMM-based sub-task boundary predictor outputs many more boundaries than the reference boundaries and thus introduces a lot of false alarms. Many boundaries are predicted because many consecutive utterances are modeled by different states. One possible cause of this problem is that the HMM states constructed by clustering similar utterances together are not accurate. Many utterances are short and do not contain enough context to robustly determine the similarity when clustering utterances that belong to the same sub-task together. Moreover, social phrases such as an acknowledgement and a yes/no response, which can occur in any sub-task, further complicate the problem. Even though the initial state labels could be adjusted by the subsequent HMM parameter estimation step which also takes into account sub-task ordering information, this step cannot help much if the initial set of states is not good enough. The clustering algorithm requires more context from larger text spans to be able to robustly identify an initial set of HMM states from data.

6.1.7.2 Topic-based segmentation

In this experiment, the bisecting *K*-means clustering algorithm inferred the initial HMM states of the HMM-based sub-task boundary predictor from data by clustering similar dialog segments together. These segments are the approximations of the true sub-tasks and were generated by the TextTiling boundary predictor, TC4DTr. This boundary predictor utilizes information from concept annotation together with regularity weights, distance weights and a triangular smoothing scheme to determine sub-task boundaries and is discussed in Section 6.1.6.6. It was chosen over other TextTiling boundary predictors because it achieved good segmentation performances in both domains. TC4DTr produced 171 dialog segments in the air travel domain. This number is slightly higher than the number of the reference segments, which is 130 segments. In the map reading domain, the number of predicted segments is lower than the number of reference segments (291 vs. 341 segments). The predicted segments provide larger context to the clustering algorithm than the utterances used in the previous experiment. The average predicted segment length is 7.8 utterances in the air travel domain and 8.6 in the map reading domain. The qualities of seeded segments in the air travel domain and the map reading domain in terms of P_k and concept-based F-measure are given in the last row of Table 6.24 and Table 6.26 respectively.

Number of states (M)	**Predicted boundaries**	P_k	**Concept-based**		
			Precision	**Recall**	**F-1**
2	4.471	0.390	0.800	0.538	0.627
4	6.112	0.392	0.747	0.632	0.667
6	6.987	0.391	0.725	0.683	0.687
8	7.562	<u>0.385</u>	0.713	0.715	<u>0.698</u>
Initial segments	8.125	0.371	0.693	0.771	0.712

Table 6.24: Segmentation performance of a topic-based HMM predictor (PC2) in the air travel domain

Table 6.24 shows segmentation performance of the topic-based HMM boundary predictor, PC2 (P stands for predicted segment), that removes pre-define stop words from the feature set. When compared to the performance of the utterance-based HMM predictor (UC2), PC2 has better overall performances both in terms of Pk and concept-based F-measure regardless of the number of HMM states used. The performance of UC2 is presented in Table 6.22. The improvement comes from the increase in precision. When the states are induced from predicted segments, the HMM predicts much fewer boundaries than when the HMM states are induced from utterances.

Nevertheless, the dialog segments obtained from the topic-based HMM predictor are not as good as the initial segments generated by the TextTiling algorithm. The HMM-based predictor output fewer boundaries and achieves slightly better precision but has lower recall. The HMM-based predictor fails to detect the boundary between two consecutive segments if both of them are assigned the same state label. Table 6.25 shows the types of the boundaries in the air travel domain that are commonly missed by the dialog segmentation algorithms. For the HMM-based segmentation, the optimal result is analyzed and the frequency of the missed boundaries is averaged from all 10 runs. The optimal result is the segmentation result that yields the best P_k and concept-based F-measure (underlined in Table 6.24) and is obtained when M is set to 8.

Left segment	**Right segment**	**Missed boundaries**	
		TextTiling (Seeding)	**HMM (PC2)**
query_flight_info	query_flight_info	14	19.2
query_flight_info	query_flights_fare	7	0.3
query_flights_fare	query_car_info	5	6.6
query_car_info	query_hotel_info	4	8.8

Table 6.25: Types and frequencies of the boundaries commonly missed by the TextTiling predictor and the HMM predictor in the air travel domain

The HMM-based predictor is able identify most of the boundaries between a **query_flight_info** sub-task and a **query_flights_fare** sub-task while the TextTiling predictor misses 31.8% of this type of boundary. Many instances of a **query_flights_fare** sub-task are quite short, only 2-3 utterances long. The TextTiling algorithm which relies on lexical cohesion has difficulty identifying two significant drops in lexical similarity of the local context that are only a couple of utterances apart as discussed in Section 6.1.6.8. The HMM-based predictor, which utilizes lexical similarity between multiple segments of dialogs in the corpus to determine topic boundaries instead of lexical cohesion, does not have this limitation.

On the other hand, the HMM-based predictor misses more boundaries between two consecutive **query_flight_info** sub-tasks as both segments are usually represented by the same state. When compared to other dialog segments in the corpus, two consecutive **query_flight_info** sub-tasks are considered more similar and are clustered into the same state in the HMM-based segmentation algorithm; however, when compare the two segments together using only local context there might be enough drop in lexical cohesion that could be considered as a sub-task boundary by the TextTiling algorithm. The HMM-based predictor also misses more boundaries between a **query_flights_fare** sub-task and a **query_car_info** sub-task, and between a **query_car_info** sub-task and a **query_hotel_info** sub-task. Some instances of those sub-tasks are clustered into the same state as they share some common concepts and keywords such as **[Fare]:**dollar and “rate”. Moreover, boundary errors in the input segments of the clustering algorithm further complicate the problem.

Number of states (M)	**Predicted boundaries**	**P_k**	**Concept-based**		
			Precision	**Recall**	**F-1**
2	11.995	0.379	0.604	0.391	0.463
4	13.660	<u>0.355</u>	0.600	0.453	<u>0.507</u>
6	14.340	0.362	0.578	0.452	0.499
8	13.690	0.369	0.581	0.433	0.486
Initial segments	15.550	0.384	0.506	0.433	0.464

Table 6.26: Segmentation performance of a topic -based HMM predictor (PC2) in the map reading domain

Table 6.26 shows segmentation performance of the topic-based HMM boundary predictor, PC2, in the map reading domain. Similar to the results in the air travel domain, when the initial HMM states are inferred from larger text spans, the overall performances both in terms of P_k and concept-based F-measure are better than the performances achieved by UC2 regardless of the number of HMM states used. The performance of UC2 is presented in Table 6.23. The improvement comes from the increase in precision. When the states are induced from predicted segments, the HMM produces much less false boundaries than when the HMM states are induced from utterances.

The segmentation result obtained from the topic-based HMM predictor is slightly better than the initial segments generated by the TextTiling algorithm. The improvement comes from the higher precision. Table 6.27 shows the number of missed boundaries for each boundary type in the map reading domain. For the HMM-based predictor, the optimal result obtained when *M* is equal to 4 is analyzed and the frequency of the missed boundaries is averaged over all 10 runs. The optimal result is underlined in Table 6.26.

Boundary type	**Missed boundaries**	
	TextTiling (Seeding)	**HMM (PC2)**
draw_a_segment	89	112.5
grounding (embedded)	117	84.9
Total	**206**	**197.4**

Table 6.27: Number of boundaries missed by the TextTiling algorithm and the HMM algorithm in the map reading domain

Even though both segmentation algorithms miss about the same number of sub-task boundaries, the types of missed boundaries are different. The HMM-based predictor misses more boundaries between two consecutive **draw_a_segment** sub-tasks as both segments are usually represented by the same state. On the other hand, the HMM-based predictor recovers more boundaries of the **grounding** sub-tasks that are embedded inside **draw_a_segment** sub-tasks while the TextTiling algorithm has difficulty with this type of boundary since the **grounding** sub-tasks are rather short. The limitation of the TextTiling algorithm in identifying the boundaries of a short sub-task is discussed in Section 6.1.6.8.

Summary

The HMM-based boundary predictor trained from predicted sub-tasks generated by the TextTiling algorithm outperforms the HMM-based predictor trained from utterances in both domains. The improvement comes from the increase in precision. When HMM states are induced from utterances, the predictor introduces a lot of false alarm boundaries since many consecutive utterances are modeled by different states. The HMM states that are induced from utterance units are not accurate because a single utterance does not contain enough context to robustly determine the similarity when clustering utterances that belong to the same sub-task together. When the bisecting *k*-means clustering algorithm induces an initial HMM states from larger text spans, predicted sub-tasks in this case, it produces a more robust state representation. The topic-based HMM boundary predictor outputs much less false boundaries and thus improves the overall segmentation performance.

While the overall segmentation quality of both dialog segmentation algorithms are not much different (the TextTiling algorithm performs slightly better in the air travel domain while the HMM-based segmentation algorithm performs slightly better in the map reading domain) the types of errors that they produce are. The HMM-based segmentation algorithm can identify more boundaries of the small sub-tasks (e.g. a **query_flights_fare** sub-task and a **grounding** sub-task) while the TextTiling algorithm, which relies on local lexical coherence, has more difficulty with short segments. On the other hand, the TextTiling algorithm can sometimes identify a boundary between two sub-task occurrences of the same type (for example, between two consecutive **query_flight_info** sub-tasks) while the HMM-based segmentation algorithm is likely to represent them by the same state; therefore, fails to detect the boundary.

6.1.7.3 Concept word representation

Since the first step of the HMM-based segmentation algorithm, the HMM state induction step, relies on a dialog segment clustering algorithm, a more efficient clustering algorithm may help improve a segmentation result. In this experiment, a concept word representation that composes of a concept label but not a concept value, the *Label* representation, is adopted. This concept word representation achieves a better sub-task clustering performance than the *Label+Word* representation, which uses both a concept label and a word string to represent a concept word, in both the air travel domain and the map reading domain as discussed in section 6.2.4.3. The *Label+Word* representation is a concept word representation used the previous experiments. Apart from a concept word representation, all other parameters used in this experiment are similar to the ones used in the PC2 boundary predictor discussed in the previous section. Table 6.28 shows the segmentation performance of the HMM-based sub-task boundary predictor that uses the *Label* representation, PL2 (L stands for the *Label* representation), in the air travel domain. The segmentation result when *M* is large is also reported since the values of both precision and recall still change as *M* increases.

Number of states (M)	Predicted boundaries	P_k	Concept-based		
			Precision	Recall	F-1
2	4.783	0.349	0.908	0.560	0.680
4	6.438	0.386	0.749	0.689	0.706
6	9.129	0.386	0.633	0.813	0.695
8	10.313	0.379	0.622	0.846	0.699
10	11.975	0.384	0.573	0.905	0.690
12	11.267	0.392	0.590	0.866	0.688
14	12.908	0.406	0.548	0.902	0.667
16	12.450	0.380	0.575	0.914	0.694
Initial segments	8.125	0.371	0.693	0.771	0.712

Table 6.28: Segmentation performance of a HMM-based boundary predictor (PL2) that uses the *label* representation in the air travel domain

When the number of HMM states increases, PL2 tends to output more boundaries. As a result, recall is higher while precision is lower; nevertheless, the overall performances both in terms of Pk and concept-based F-1 do not change much. The effect of the number of HMM states on segmentation performance is quite similar to the one found in the result of PC2 when the *Label+Word* representation is used. However, the amount of the change in both precision and recall is larger in PL2.

Figure 6.13 presents a comparison between the two HMM-based predictors that use different concept word representations, PC2 and PL2. The graphs for PC2 are in darker colors while the graphs for PL2 are in lighter colors. PL2 has much higher recall than PC2 when M is larger but has much lower precision due to the higher number of the boundaries predicted. In PL2, more utterances are reassigned to different states during several iterations of the Viterbi decoding and HMM parameter re-estimation. When the *Label+Word* representation is used, the similarity between concept words is constrained by both the concept label and their values; therefore, utterances are less likely to be reassigned to different states. As a result, PL2 outputs more boundaries than PC2; nevertheless, concept-based F-1 of both predictors are about the same. When compared to the quality of the initial segments generated by the TextTiling algorithm shown in the last row of Table 6.28, the PL2 predictor does not produce a result that has a better overall quality.

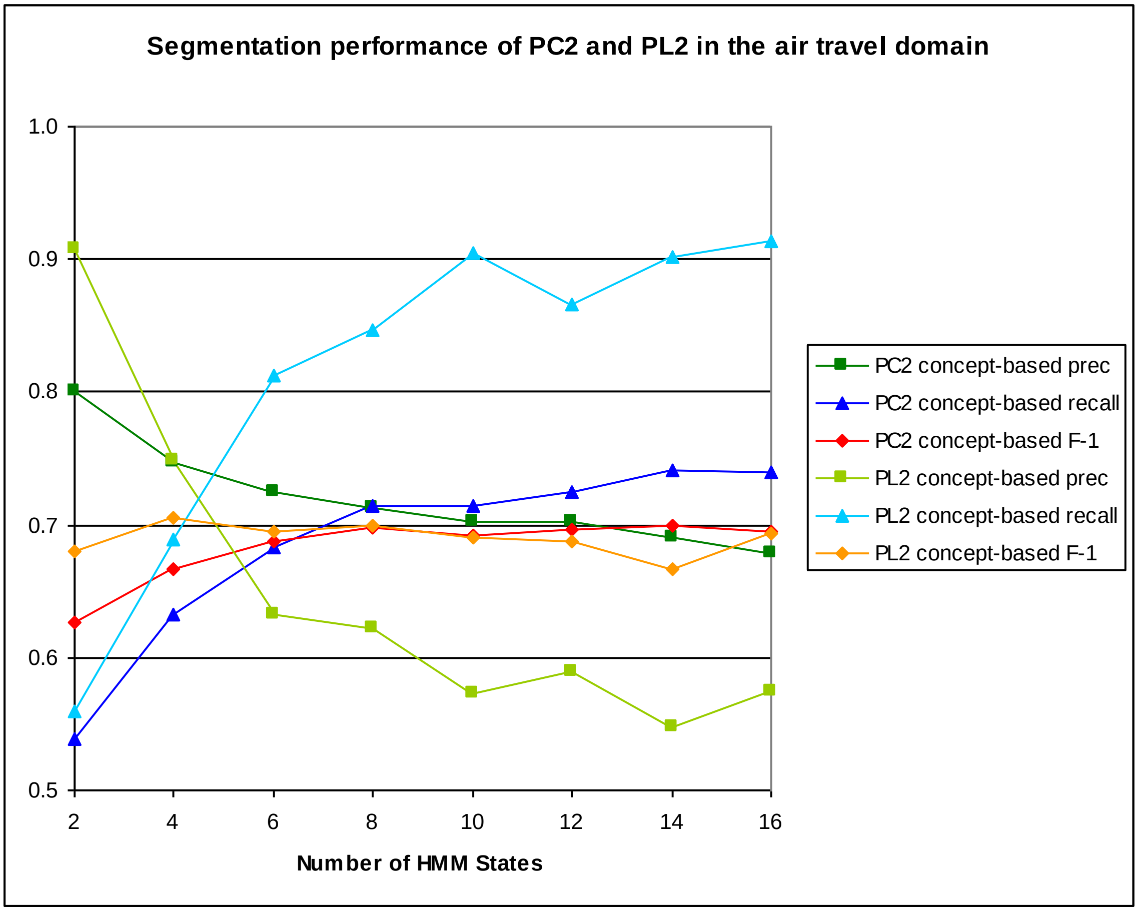


Figure 6.13: A performance comparison between two sub-task boundary predictors that use different concept word representations in the air travel domain

Table 6.29 shows segmentation performance of the HMM-based sub-task boundary predictor that uses the *Label* representation, PL2, in the map reading domain. When the number of HMM states increases, PL2 predicts more boundaries and achieves higher recall, but precision and overall performances are lower both in terms of P_k and concept-based F-1.

Number of states (M)	Predicted boundaries	P_k	Concept-based		
			Precision	Recall	F-1
2	13.035	$\underline{0.250}$	0.864	0.592	$\underline{0.686}$
4	27.405	0.306	0.522	0.775	0.619
6	30.515	0.327	0.494	0.818	0.612
8	32.715	0.335	0.471	0.829	0.596
Initial segments	15.550	0.384	0.506	0.433	0.464
PC2 (Label+Word)	13.660	0.355	0.600	0.453	0.507

Table 6.29: Segmentation performance of a HMM-based boundary predictor (PL2) that uses the *label* representation in the map reading domain

Figure 6.14 presents a comparison between the two HMM-based predictors that use different concept word representations, PC2 and PL2. The graphs for PC2 are in darker colors while the graphs for PL2 are in lighter colors. PL2, which uses the *Label* representation, achieves better overall performances both in terms of P_k and concept-based F-measure regardless of the number of HMM states used. The improvement comes mainly from higher recall. Similar to the observation found in the air travel domain, PL2 predicts more boundaries than PC2 after several iterations of the Viterbi decoding and HMM parameter re-estimation as more utterances are reassigned to different states.

The optimal result of the boundary predictor that uses the *Label* representation (PL2) is also better than the optimal result of the predictor that uses the *Label+Word* representation (PC2) on all evaluation metrics. The optimal result of PL2 is obtained when M is equal to 2 and is underlined in Table 6.29 while the optimal result of PC2 is obtained when M is equal to 4 and is shown in the last row of the same table. The improvement obtained from a better concept word representation is statistically significant both in terms of P_k (p-value = 0.0006) and concept-based F-measure (p-value = 0.0004). The segmentation result of PL2 is also significantly better than the result obtained from the TextTiling algorithm shown in the 2nd to last row of Table 6.29. The p-value for the difference in P_k is 0.0005 and the p-value for the difference in concept-based F-measure is 0.0003.

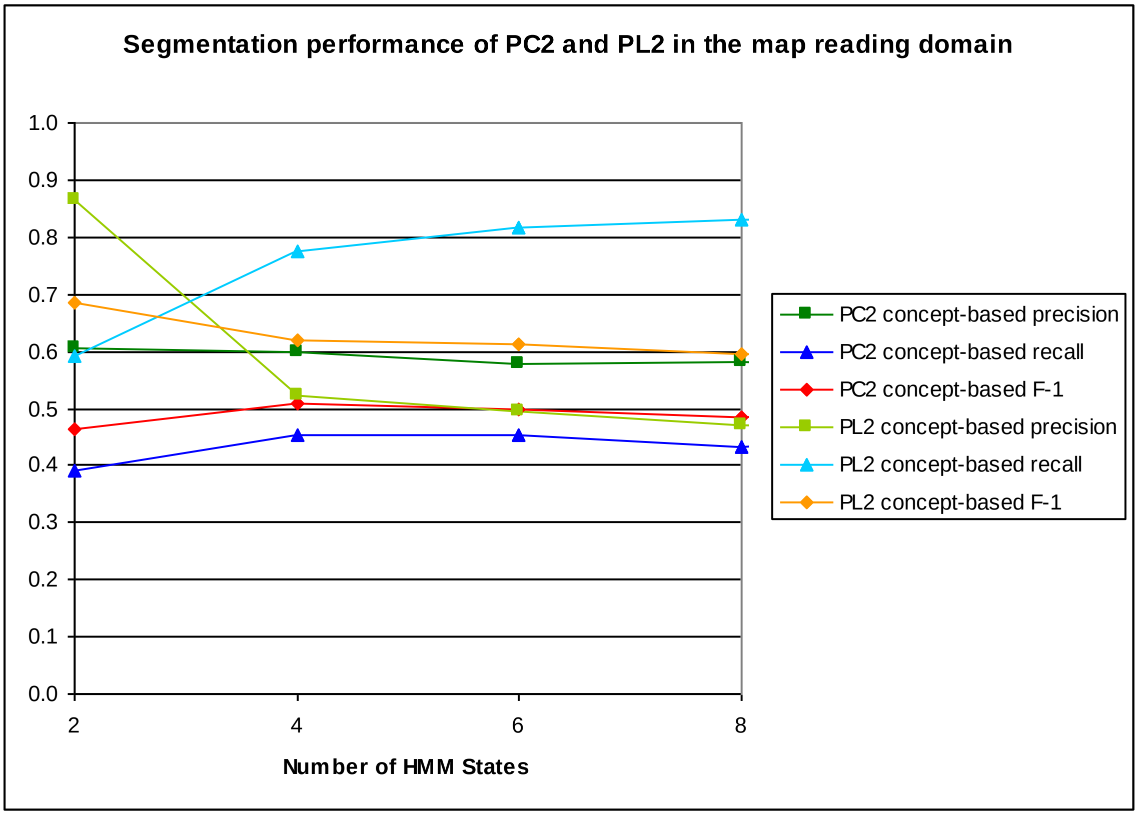


Figure 6.14: A performance comparison between two sub-task boundary predictors that use different concept word representations in the map reading domain

Boundary type	Missed boundaries	
	PC2	**PL2**
draw_a_segment	112.5	111.9
grounding (embedded)	84.9	28.5

Table 6.30: Types and number of boundaries missed by the HMM-based predictors that use different concept word representations in the map reading domain

The types of boundaries in the map reading domain missed PC2 and PL2 are shown in Table 6.30 along with their frequencies. When compare between two HMM-based predictors that uses different concept word representations, PL2 can recover many more boundaries of the **grounding** sub-tasks embedded inside the **draw_a_segment** sub-tasks than PC2. In PL2, dialog segments that belong to different sub-tasks are well separated into different HMM states since the *Label* representation can distinguish between the two sub-tasks with higher accuracy than the *Label+Word* representation as discussed in section 6.2.4.3.

Summary

When the *Label* representation, which is a more suitable concept word representation for sub-task clustering, is adopted in the HMM-based segmentation algorithm, a better segmentation result can be achieved in the map reading domain. The performance gain is statistically significant when compared to both the HMM-based segmentation algorithm that uses the *Label+Word* representation and the TextTiling algorithm. However, in the air travel domain, where the clustering result is marginal better when the *Label* representation is used, there is no difference in the overall segmentation performance between the HMM-based sub-task boundary predictors that uses different concept word representations.

The HMM-based segmentation algorithm that uses the *Label* representation allows more utterances to be reassigned to different HMM states during the Viterbi decoding and HMM parameter re-estimation iterations. Therefore, it can discover more sub-task boundaries than the predictor that uses the *Label+Word* representation. Nevertheless, the number of false alarms is also higher.

6.1.7.4 Stop word treatment

It has been shown in the TextTiling experiments discussed in Section 6.1.6 that a data-driven stop word treatment, which determines a list of stop words dynamically from word distribution is each data set, can improve segmentation results especially in the air travel domain. In this experiment four HMM-based sub-task boundary predictors that utilize different stop word treatments are investigated. PL1 is a boundary predictor that uses all tokens in the transcription as features. PL2 is a boundary predictor that removes stop words defined manually from a set of features and is the same predictor as the one discussed in the previous experiment. PL3 is a boundary predictor that removes data-driven stop words from a set of features. PL4 is a boundary predictor that weighs each word in the transcription with a regularity weight computed by Equation (6.1).

For PL3 and PL4, a regularity count for a concept word is computed from an average regularity count of all of the words that belong to the same concept type since those words are represented by the same token in the *Label* representation. In this experiment, the threshold for selecting data-driven stop words was set to $\mu + 2*\sigma$; where μ is the mean of the regularity counts of all of the words in a given corpus and σ is their standard deviation. A data-driven stop word list contains 24 words in the air travel domain and 21 words in the map reading domain while a hand-crafted stop word list contains 174 words[10].

For PL4, regularity weights are only used when calculating the cosine similarity between dialog segments in the HMM state induction step but not when modeling the state emission probabilities in the HMM parameter estimation step. A regularity weight, while reflecting the significance of each word when determining the similarity between two dialog segments, does not indicate a likelihood of a word being generated from a given state. To handle common words when creating state-specific language models, the PL4 predictor excludes data-driven stop words from the vocabulary similar to the PL3 predictor.

Figure 6.15 shows the performance of each sub-task boundary predictor in the air travel domain in terms of concept-based F-1 when the number of HMM states (*M*) increases. All of the predictors output more boundaries when the models contain more HMM states. As a result, precision tends to decreases while recall tends to increase. For PL2, PL3 and PL4, the overall performance in terms of concept-based F-1 does not change much as the loss in precision can be compensated by the gain in recall. However, for PL1, concept-based F-1 slightly decreases as *M* increases. The performance of PL1, which has no special treatment for common words, is slightly lower than other predictors at all values of *M*. PL1 predicts more sub-task boundaries and creates more false alarms than other predictors. Some of the false alarms are correlated with sub-structures within a **query_flight_info** sub-task such as an extra boundary between a dialog segment that specifies all of the criteria for retrieving flight information and a dialog segment that discusses the results retrieved from the database.

[10] A hand-crafted stop word list was taken from the list that was created as part of the Snowball project http://search.cpan.org/~creamyg/Lingua-StopWords-0.08/lib/Lingua/StopWords.pm.

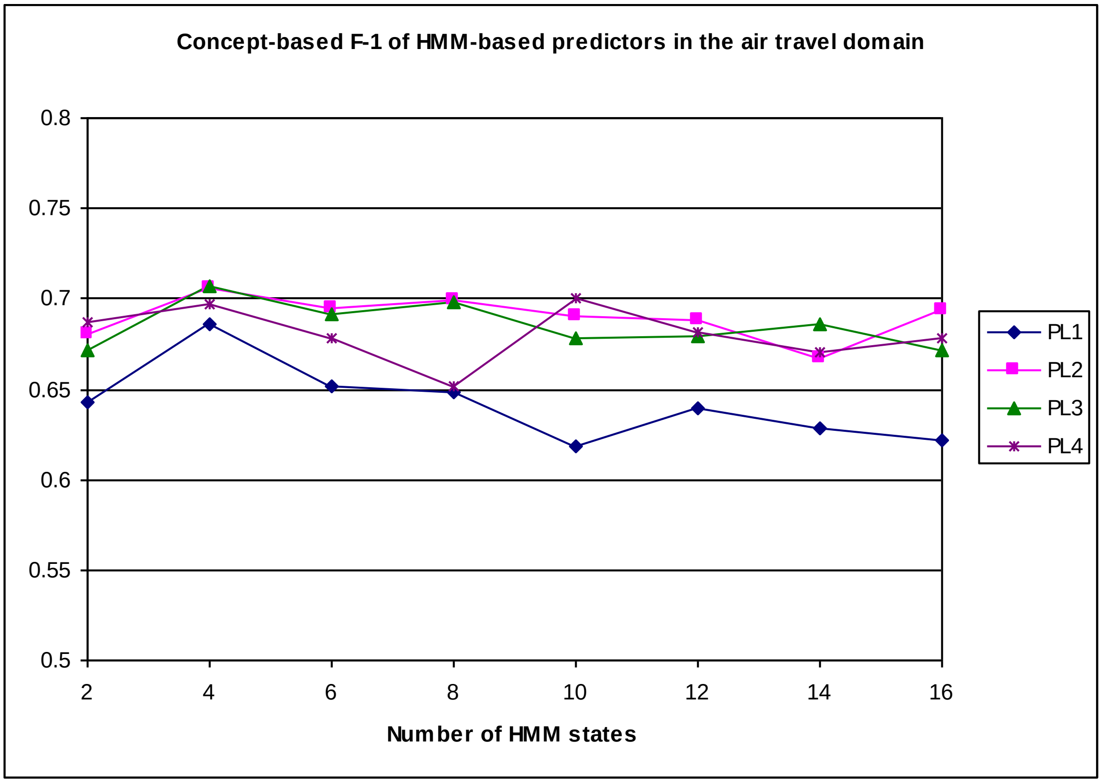


Figure 6.15: Concept-based F-measure of HMM-based sub-task boundary predictors that use different stop word treatments in the air travel domain

Predictor	Number of states (Optimal)	Predicted boundaries	P_k	Concept-based		
				Precision	Recall	F-1
PL1	4	8.621	0.393	0.646	0.770	0.686
PL2	4	6.438	0.386	0.749	0.689	0.706
PL3	4	7.971	0.380	0.685	0.761	0.707
PL4	10	10.696	0.376	0.606	0.874	0.700
Initial segments	-	8.125	0.371	0.693	0.771	0.712

Table 6.31: Optimal performances of the HMM-based boundary predictors when different stop word treatments are applied in the air travel domain

The best performance of each sub-task boundary predictor is shown in Table 6.31. The performance of PL1 is slightly worse than those of other predictors; however, the differences are not statistically significant. When compared to the quality of the initial segments generated by the TextTiling algorithm shown in the last row of Table 6.31, none of the HMM-based segmentation algorithms produces a result that has a better overall quality.

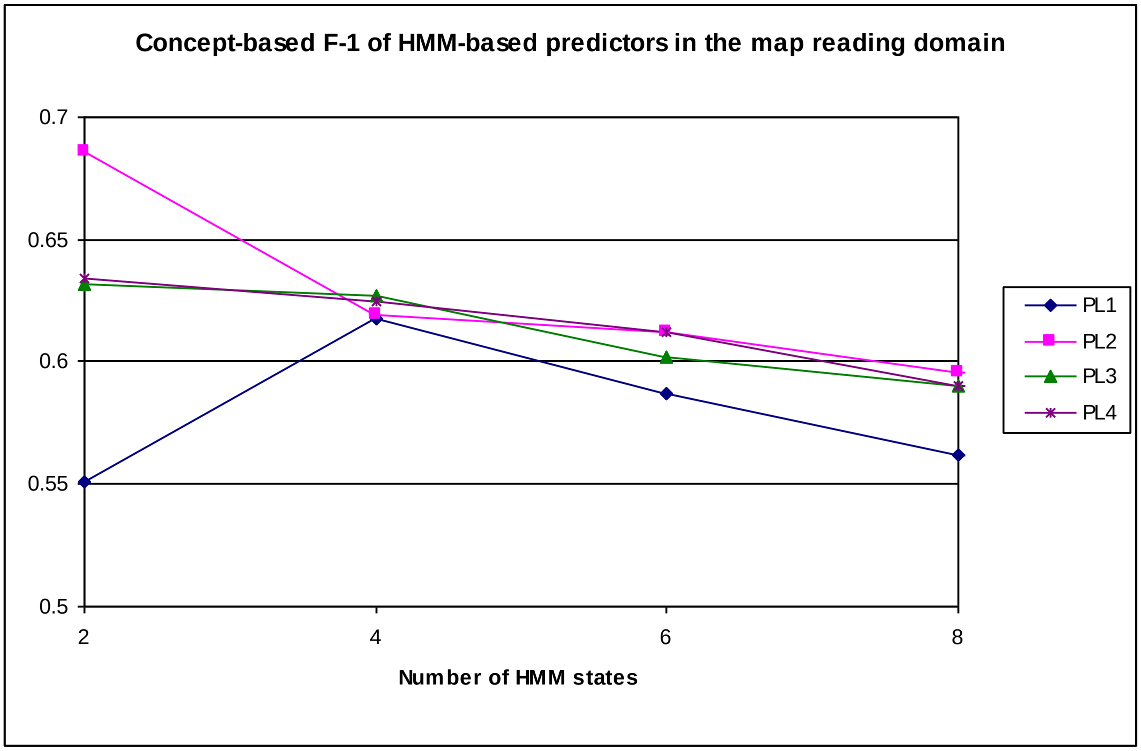


Figure 6.16: Concept-based F-measure of HMM-based sub-task boundary predictors that use different stop word treatments in the map reading domain

Figure 6.16 shows the performances of the HMM-based sub-task boundary predictors that use different stop word treatments in the map reading domain. Similar to the result obtained in the air travel domain, the performance of PL1, which has no special treatment for common words, is lower than other predictors at all values of *M*. PL1 predicts more sub-task boundaries and creates more false alarms than other predictors. When M is equal to 2 the performances of all four sub-task boundary predictors are substantial different. The difference stems from the ability of the HMM in distinguishing between a **draw_a_segment** sub-task and a **grounding** sub-task. In PL2, where both sub-tasks are well separated into two different states, the segmentation performance is higher as the predictor can recover more boundaries of the **grounding** sub-tasks. PL3 and PL4 can also distinguish between the two sub-tasks, but not in every run; therefore, their average performances are lower. In PL3 and PL4, two keywords “go” and “got” which occur in a **draw_a_segment** sub-task and a **grounding** sub-task respectively are removed by data-driven stop word treatments as they occur regularly throughout the dialogs. These two keywords can help distinguishing between the two sub-tasks in additional to the concept words.

Predictor	Number of states (Optimal)	Predicted boundaries	P_k	Concept-based		
				Precision	Recall	F-1
PL1	4	27.59	0.308	0.521	0.778	0.618
PL2	2	13.035	**0.250**	0.864	0.592	**0.686**
PL3	2	12.490	0.281	0.829	0.541	0.632
PL4	2	12.500	0.280	0.830	0.543	0.634
Initial segments	-	15.550	0.384	0.506	0.433	0.464

Table 6.32: Optimal performances of the HMM-based boundary predictors when different stop word treatments are applied in the map reading domain

The best performance of each sub-task boundary predictor is shown in Table 6.32. The overall performances of PL2 are significantly better than the performance of other predictors both in terms of Pk and concept-based F-measure at a significance level of 0.005 (p-value < 0.005). When compared to the quality of the initial segments generated by the TextTiling algorithm shown in the last row of Table 6.32, all HMM-based segmentation algorithms produce significantly better segmentation results in terms of both Pk and concept-based F-measure at a significance level of 0.05 (p-value < 0.05).

Summary

In the air travel domain, the choice of a stop word treatment is less crucial for the HMM-based segmentation algorithm than for the TextTiling algorithm. However, in the map reading domain, the segmentation results obtained from the sub-task boundary predictors that utilize different stop word treatments are significantly different. The predictor that removes the stop words manually defined outperforms other predictors both in terms of both P_k and concept-based F-measure.

Both data-driven approaches (a data-driven stop word list and a regularity weight), which are developed based on lexical coherence assumption, are suitable for the TextTiling algorithm that follows the same assumption but not for the HMM-based segmentation algorithm which relies instead on recurring patterns in the corpus. The data-driven approach may remove some keywords that can distinguish among different types of sub-tasks if they occur regularly throughout the dialogs. As a result, dialog segments from different types of sub-tasks are represented by the same HMM state which makes the predictor misses the boundaries between those sub-tasks. Nevertheless, the HMM-based sub-task boundary predictors that utilize data-driven stop word treatments achieve slightly better segmentation performances than the predictor that has no special treatment for common words in both domains.

6.1.8 Discussion and conclusion

In the previous sections, two unsupervised discourse segmentation approaches, a TextTiling algorithm and a hidden Markov model, have been investigated through a series of experiments. In this section, I will discuss some interesting findings and then summarize the results from all of the experiments.

The proposed data-driven stop word treatments can discover stop words that are specific to the given domain and genre. While a performance of the sub-task boundary predictor that uses a hand-crafted stop word list depended on compatibility between the pre-selected word list and the domain, the performance of the boundary predictors that use data-driven stop word treatments (a dialog-specific stop word list and a regularity weight) are not since the stop words are determined directly from word distribution in each data set. However, the efficiency of the data-driven stop word treatments is also depended on word distribution characteristic. If the distributions of content words are quite similar to those of stop words, the data-driven approaches are less efficient as in the map reading domain. The proposed data-driven stop word treatments are developed based on lexical coherence assumption; therefore, these treatments are effective when used by the TextTiling algorithm, which follows the same assumption, but are not as effective when used by the HMM-based segmentation algorithm, which relies instead on recurring patterns in the corpus. Nevertheless, the HMM-based predictors that utilize data-driven stop word treatments achieve slightly better segmentation performances than the predictor that has no special treatment for common words in both domains.

Information from concept annotation can improve dialog segmentation results. It provides a richer representation that helps distinguish between different concept types and also between different word senses. Co-occurrences of the same word string that actually belong to dissimilar concepts or dissimilar worse senses no longer affect the similarity score. In addition, this better representation makes the distributions of content words more distinguishable from the distributions of stop words; therefore, the boundary predictors that use data-driven stop word treatments, which rely on word distribution characteristic, can achieve more performance improvement. In the domain that the distributions of contents words and the distributions of stop words are quite similar, such as the map reading domain, the predictor that gives common words small weights rather than removing them from a set of features achieves a better performance.

The size of context windows is a crucial parameter in the TextTiling algorithm. A small context window, which is more sensitive to small changes in the context, is more appropriate for identifying fine-grained segments in the task-oriented dialogs particularly when segment length variation is high. Two modifications to the standard TextTiling algorithm are proposed to specifically handle fine-grained segments. *Distance weights*, which demote the similarity between far away contexts, improve the overall segmentation performance in most conditions while a *triangular smoothing scheme* produces a more precise boundary prediction. The segmentation results obtained from the modified TextTiling algorithm are significantly better than baselines in both domains.

A cut-off threshold for selecting a list of boundaries from all candidate boundaries given their depth scores is another important parameter. An appropriate cut-off threshold is depended on the characteristic of the domain. If consecutive sub-tasks in a dialog are quite similar, a lower threshold is required as in the map reading domain. Similarly, a domain which contains short sub-tasks may also require a low cut-off threshold.

For the HMM-based segmentation algorithm, the sub-task boundary predictor trained from predicted sub-tasks generated by the TextTiling algorithm outperforms the HMM-based predictor trained from utterances in both domains. A single utterance does not contain enough context to robustly determine the similarity when clustering utterances that belong to the same sub-task together; therefore, the HMM states that are induced from utterance units are not accurate. Predicted segments provide more context to the clustering algorithm that induces the initial set of HMM states. Thus, the clustering algorithm produces a more robust state representation. The use of predicted segments also eliminates the need of annotated data that would be required if the HMM states are induced from reference segments.

A more efficient clustering algorithm can also improve the performance of the HMM-based segmentation algorithm since it provides a better state representation that can differentiate between dialog segments that belong to dissimilar sub-tasks. When the *Label* representation which is a more suitable concept word representation for sub-task clustering than the *Label+Word* representation is used, a HMM-based sub-task boundary predictor produces a better segmentation result. In the map reading domain where the *Label* representation makes dialog segments from different sub-tasks much more distinguishable, a significant improvement can be achieved.

The types of errors produced by the TextTiling algorithm and the HMM-based segmentation algorithm are different. In both domains, the HMM-based segmentation algorithm can identify more boundaries of small sub-tasks, such as a **query_flights_fare** sub-task and a **grounding** sub-task, than the TextTiling algorithm. The TextTiling algorithm, which relies on local lexical cohesion, is unlikely to find two significant drops in lexical similarity that are only a couple of utterances apart and thus fails to detect a boundary of a short segment. The HMM-based predictor, which utilizes lexical similarity between multiple segments of dialogs in the corpus to determine topic boundaries instead of lexical cohesion, does not have this limitation. On the other hand, the HMM-based segmentation algorithm misses more boundaries between two sub-task occurrences of the same type (for example, between two consecutive **query_flight_info** sub-tasks and two consecutive **draw_a_segment** sub-tasks) as they are usually represented by the same state. When compared to other dialog segments in the corpus, two consecutive sub-task of the same type are usually more similar and are clustered into the same state by the HMM-based segmentation algorithm; however, when compare the two segments together using only local context there might be enough drop in lexical cohesion that could be considered as a sub-task boundary by the TextTiling algorithm.

Table 6.33 summarizes the best dialog segmentation results from the TextTiling algorithm and the HMM-based segmentation algorithm. The best result of the TextTiling algorithm is obtained when the sub-task boundary predictor utilizes information from concept annotation together with regularity weights, distance weights and a triangular smoothing scheme. For the HMM-based segmentation, the best result is obtained when the predictor utilizes the *Label* representation together with the hand-crafted stop word list. The optimal number of HMM states is 4 for the air travel domain and 2 for the map reading domain. Both predictors produce good segmentation results in both domains. Nevertheless, please note that some variations of the TextTiling algorithm and the HMM-based segmentation algorithm do achieve better segmentation performances than the ones presented in Table 6.33 but only for one domain.

Algorithm	**Air Travel**		**Map Task**	
	P_k	**C. F-1**	**P_k**	**C. F-1**
TextTiling	0.371	0.712	0.384	0.464
HMM-based	0.386	0.706	0.250	0.686

Table 6.33: The best segmentation performances of the TextTiling algorithm and the HMM-based algorithm in terms of P_k and concept-based F-1.

Both text segmentation approaches while performing well with expository text require some modifications when applied to spoken dialogs and a fine-grained segmentation problem. For the TextTiling algorithm, the proposed modifications, which include a data-driven stop word treatment, a distance weight, and a triangular smoothing scheme, significantly improve segmentation performance over the baseline TextTiling algorithm. For the HMM-based segmentation algorithm, the use of larger text spans when inducing HMM states and a more abstract concept word representation help improve the results, especially on the map reading domain. When comparing between the two algorithms, the HMM-based segmentation algorithm performs significantly better than the TextTiling algorithm in the map reading domain since the HMM-based segmentation algorithm can identify more boundaries of small sub-tasks which occur quite often in the map reading domain. The TextTiling algorithm, on the other hand, performs better in identifying the boundaries between consecutive sub-tasks of the same type. Since both dialog segmentation algorithms produce error types that complement each other, the segmentation result could be improved by combining both segmentation approaches together.

Algorithm	**Air Travel**		**Map Task**	
	Utterance-based P_k	**F-1**	**Utterance-based P_k**	**F-1**
TextTiling	0.427	0.457	0.431	0.292
HMM-based	0.417	0.456	0.265	0.552

Table 6.34: The best segmentation performances of the TextTiling algorithm and the HMM-based algorithm in terms of utterance-based P_k and standard F-measure

The performances of the best dialog segmentation algorithms in Table 6.33 are presented again but with different performance metrics. In Table 6.34 the performances of the segmentation algorithms are reported in terms of utterance-based P_k, (the unit of k is an utterance) and standard F-measure in order to make a direct comparison with the performance of the segmentation algorithms developed by other researchers. Since segment granularity is one factor that affects segmentation performance (Arguello and Rosé, 2006), the performances of the dialog segmentation algorithms proposed in this chapter are compared to the performances of the segmentation algorithms that were used to identify the boundaries of fine-grained segments. Arguello and Rosé (2006) reported the performances of several dialog segmentation algorithms. One of their corpora, the Thermo corpus, contains fine-grained segments of topics in tutoring dialogs where the topic is defined based on a discourse segment purpose. The average length of the segments in this corpus is 13.3 utterances (68.1 words) which is about the same as the average sub-task length in the air travel domain and the map reading domain. The best segmentation result on the Thermo corpus (utterance-based P_k = 0.404, F-1 = 0.369) was obtained from a supervised segmentation algorithm, a Naïve Bayes classifier trained on several textual features and prosodic features. The best segmentation performances achieved by the unsupervised segmentation algorithms proposed in this thesis are comparable to their result.

Both unsupervised dialog segmentation algorithms, TextTiling and HMM-based segmentation, can identify the boundaries between sub-tasks with acceptable accuracy. Nevertheless, both of algorithms, which rely on lexical similarity between dialog segments, have some difficulty identifying the boundary between two sub-tasks of the same type. Lexical similarity, which is an efficient feature for identifying the boundary between two sub-tasks that belong to dissimilar form types, may not provide enough information for determining the boundary between two instances of the same form type since their contents are more similar. To solve this problem, additional features that are not sensitive to the content of the segment, such as a prosodic feature, are necessary. It has been shown that prosodic features are able to identify a boundary between segments that contain quite similar contents (Swerts and Ostendorf, 1997).

6.2 Sub-task clustering

After segmenting all in-domain dialogs into sequences of sub-tasks, the next step in form identification is to group the dialog segments that belong to the same type of sub-task together as they represent the same form type, i.e. dialog segments that correspond to a **query_flight_info** sub-task are grouped together in one cluster while dialog segments that correspond to a **query_hotel_info** sub-task are grouped together into another cluster. Sub-task clustering can be considered as one type of document clustering where a document is equivalent to a dialog segment which represents a sub-task in the form-based dialog structure representation.

Similar to other learning problems, approaches for document clustering can be classified into two main categories: supervised approach and unsupervised approach. Supervised document clustering requires a set of pre-defined categories and also labeled data for training. K-Nearest Neighbor, Naive Bayes and Support Vector Machines are among many well-known supervised document clustering algorithms that have been applied to the problem of document classification or document categorization. Detail discussion of these algorithms and other supervised clustering algorithms along with their performance comparison can be found in survey literature (Aas and Eikvil, 1999; Sebastiani, 2002; Yang and Liu, 1999).

Unsupervised clustering approaches, on the other hand, rely on context similarity between two text segments instead of labeled data. These approaches follow the assumption that two text segments that belong to the same group are more similar than two text segments that belong to different groups. Since no pre-defined category is required, an unsupervised document clustering approach has been used to explore the structure of a document collection (Cutting et al., 1992). This problem is quite similar to a sub-task clustering problem where we would like to discover the structure of dialogs in a new domain and that a pre-defined list of sub-tasks is not available.

Unsupervised clustering approaches can be categorized into hierarchical clustering and non-hierarchical clustering (or partitional clustering). A hierarchical clustering approach produces a nested partitions of documents in a tree-like structure with a root cluster at the top of the tree contains all of the documents while leaf nodes at the bottom of the tree contain only a single document. Hierarchical clusters can be generated in either a top-down or a bottom-up fashion. A divisive hierarchical clustering approach (or a top-down approach) starts with one cluster that contains all of the documents and, at each step, splits a cluster until all of the clusters contain only a single document or the desired number of clusters is obtained. An agglomerative hierarchical clustering approach (or a bottom-up approach), works on the opposite direction, starts with single document clusters and, at each step, merges the most similar clusters together until all of the documents are merged into one cluster or until the desired number of clusters is obtained. In contrast to a hierarchical clustering approach, a partitional clustering approach creates all clusters at once by partitioning the data into *K* groups where *K* is the number of desired clusters. More detail discussion on unsupervised document clustering approaches can be found in (Rasmussen, 1992).

A hierarchical clustering approach is known for its quality while a partitional clustering approach is known for its efficiency (Larsen and Aone, 1999; Steinbach et al., 2000). To combine the strength of both techniques many hybrid approaches have been proposed (Cutting et al., 1992; Larsen and Aone, 1999; Steinbach et al., 2000). In this section, the bisecting *K*-means algorithm which is a hybrid approach proposed by Steinbach et al. (2000) was chosen as a sub-task clustering algorithm since it has been shown to produce as good or better results than an agglomerative hierarchical clustering approach and also have more efficient run time.

In the following sections, Section 6.2.1, I first discuss the features used in sub-task clustering and their representations. Then in Section 6.2.2 a sub-task clustering algorithm, the bisecting *K*-means algorithm, is described. The clustering results were evaluated with the evaluation metrics discussed in Section 6.2.3. The experiments and the clustering results are given in Section 6.2.4. Finally, all the findings are concluded in Section 6.2.5.

6.2.1 Feature representation

Sub-task clustering features are taken from dialog transcription. Each transcribed word is pre-processed by removing morphological inflections using the Porter's stemming algorithm. If a set of domain concepts has already been identified, information from concept annotation can also be utilized. In dialog segmentation algorithms discussed in Section 6.1.1.1 both a concept label and its value are used together to represent a concept word. Examples of this joint representation are **[DepartureCity]:**pittsburgh and **[hour]:**one.

However, for a sub-task clustering problem, based on assumption that a list of concepts occurs in one sub-task is distinguishable from a list of concepts occurs in other sub-tasks regardless of the values of the concepts, a concept label is more informative than its value. From this assumption a more abstract representation which focuses on a concept type rather than its value may be a more suitable representation. Hence, in stead of representing a concept word with both a concept label and a concept value, only a concept label is used to make the representation generalized over all different values of the same concept type. For example, **[ArrivalCity]:**pittsburgh and **[ArrivalCity]:**boston are represented with the same token **[ArrivalCity]**. This new representation that uses only a concept label to represent a concept word is referred to as the *Label* representation while a concept word representation that uses both a concept label and a word string to represents a concept word is referred to as the *Label+Word* representation.

6.2.2 Bisecting K-means clustering algorithm

The bisecting *K*-means algorithm (Steinbach et al., 2000). is a top-down clustering algorithm that combines a hierarchical clustering approach and a partitioning approach together by applying the standard *K*-means clustering (a partitional clustering) repeatedly to create hierarchical clusters. The algorithm utilizes cosine similarity between dialog segments in order to assign the segments into clusters. The algorithm starts with a single cluster that contains all of the dialog segments in the corpus. Then, at each iteration the largest cluster is split into two sub-clusters until the desired number of clusters is reached. The *bisecting* algorithm, which splits one cluster into two clusters, is as follows:

1. Choose two dialog segments from the cluster randomly and use them as the initial centroids of the new sub-clusters. Each dialog segment is represented by a vector; each dimension of the vector represents the frequency of each token in the segment similar to the context vector used in the TextTiling algorithm discussed in Section 6.1.2.

2. For each dialog segment in the original cluster, calculate the cosine similarity between the segment and each centroid of the new sub-clusters. Assign the dialog segment to the cluster of the most similar centroid.
3. Re-compute the centroid of each new sub-cluster
4. Repeat step 2 and 3 until the assignment in step 2 is stable (the number of changes in cluster assignment is less than a pre-defined threshold, *threshold-1*)

This algorithm is similar to the standard *K*-means algorithm when *K* is set to 2. The algorithm is repeated for several times (*B* times) and the split that produces the highest overall similarity is taken. For more detail discussion of the bisecting *K*-means algorithm, please refer to (Steinbach et al., 2000). Each cluster in a set of clusters outputted by the bisecting *K*-means algorithm corresponds to a sub-task. The algorithm also creates an etcetera state by combining small clusters (i.e. the clusters that contain less than *T* segments) together to capture the sub-dialogs that are not relevant to the task.

Since each dialog segment in the same cluster is an instance of the same sub-task, it is associated with the same form type. Therefore, by simply extracting a list of concepts contained in each cluster, a set of slots that is associated with each type of form can be determined.

6.2.3 Evaluation metrics

To evaluate the performance of each sub-task clustering algorithm, the output clusters are compared against a set of reference sub-tasks in a domain of interest using the same set of evaluation metrics used to assess the performance of a concept clustering algorithm. The metrics include *precision*, *recall* and *singularity score* and are described in detail in Section 5.3. Equation (5.7), (5.8) and (5.9) can also be used to compute precision, recall and singularity score for each sub-task respectively when the notion of items in a cluster is changed from words to dialog segments and R_i is referred to a reference sub-task instead of a reference concept.

To compare the output clusters with a set of reference sub-tasks, first, a mapping between each cluster and its corresponding sub-task has to be created. Similar to the evaluation of concept clustering, a many-to-one mapping between multiple clusters and a reference sub-task is allowed. Since the clustering algorithm does not assign a sub-task label to each cluster, a majority voting scheme is used to identify the sub-task that each cluster represents. Firstly, a sub-task label is assigned to each dialog segment in a cluster. Then, the sub-task label that occurs most often in the cluster, or the *majority sub-task*, is assigned as a sub-task label for that cluster. It is quite straight forward to identify a concept label for each word in a cluster when a list of concept members of each reference concept is given. However, determining a sub-task label for each dialog segment in a cluster is more complicate when dialog segments are obtained from an automatic segmentation algorithm as these segments may have inaccurate boundaries. A dialog segment may not correspond to a single sub-task. Moreover, the number of predicted segments and the number of reference segments (or sub-tasks) may not be the same which make it more difficult to align the predicted segments with the reference sub-tasks.

Figure 6.17 illustrates the alignments between reference sub-tasks and predicted segments. In Figure 6.17 (a), since the boundary between segment "a" and segment "b" is not accurate, segment "a" corresponds to both sub-task "A" and sub-task "B". The alignment can be more complicated when the number of predicted segments is different from the number of reference sub-tasks as shown in Figure 6.17 (b) and Figure 6.17 (c). In Figure 6.17 (b) where there are fewer predicted segments than reference segments, segment "i" corresponds to both sub-task "I" and sub-task "J". On the other hand, when there are more predicted segments than reference segments as in Figure 6.17 (c), multiple segments, "y" and "y′", correspond to a single sub-task "Y". The latter case is also problematic since the unit of the numerator in Equation (5.8) is no longer the same as the unit of the denominator (a predicted segment versus a reference segment).

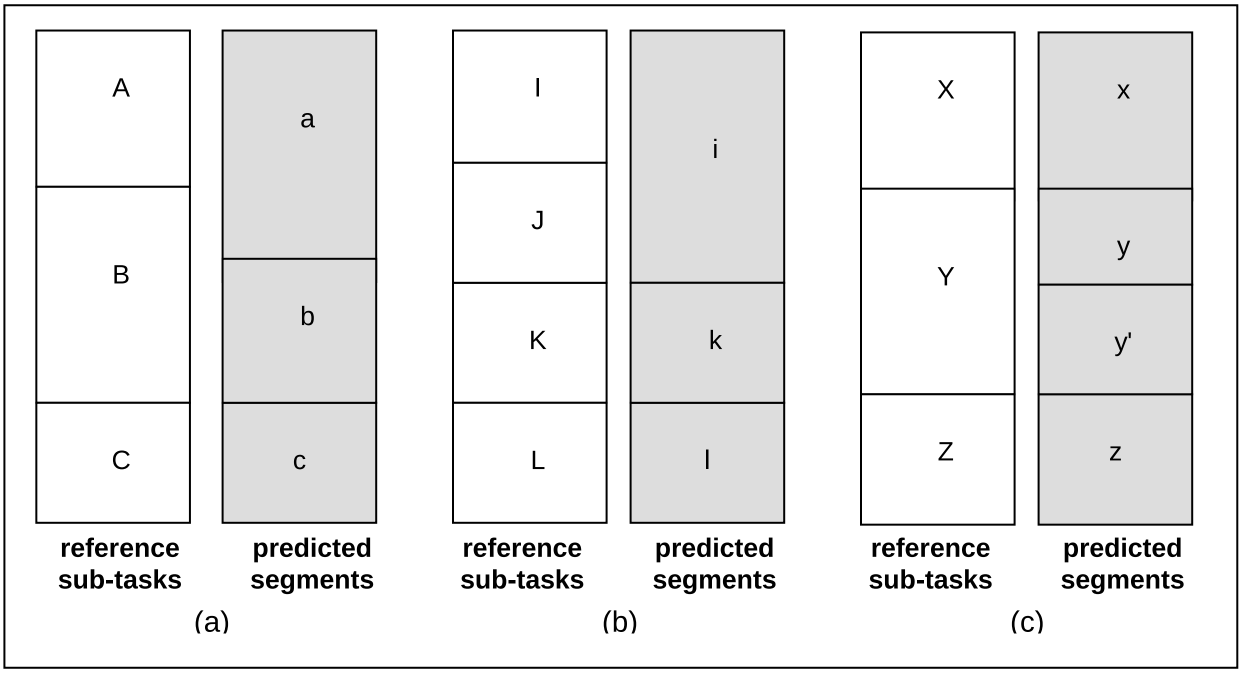


Figure 6.17: Sample alignments between reference sub-tasks and predicted segments

To avoid the complication in aligning predicted segments with reference sub-tasks, I chose to work on a smaller and unambiguous unit, namely an utterance. It is straight forward to assign a sub-task label to each utterance of dialog segments in a cluster. A majority sub-task can be defined as a sub-task that encompasses the greatest number of utterances in a cluster. However, this voting scheme favors a long sub-task. To resolve this problem each utterance is assigned a weighted count that is inversely proportional to the length of the sub-task that it belongs to. Specifically, a *weighted utterance count* for each utterance is set to $\frac{1}{n}$; where *n* is the length of the corresponding sub-task. Therefore, a majority sub-task is redefined as a sub-task that encompasses the greatest amount of weighted utterance counts. By using this weighting scheme, the units of both the numerator and the denominator in Equation (5.8) remain the same, a reference segment.

Equation (5.7) and (5.8) are presented again in Equation (6.12) and (6.13) respectively with the new notions that make the equations more suitable for calculating the precision and recall of a given sub-task. Specifically, let R_i be a reference sub-task of interest and C_1, C_2, …, C_{m_i} be the clusters that represent the sub-task R_i; where m_i is the number of the clusters. The precision and recall of the sub-task R_i are as follows:

$$precision(R_i) = \frac{\sum_{j=1}^{m_i} weighted\ utterance\ counts\ in\ C_j\ that\ belong\ to\ R_i}{\sum_{j=1}^{m_i} weighted\ utterance\ counts\ in\ C_j} \qquad (6.12)$$

$$recall(R_i) = \frac{\sum_{j=1}^{m_i} weighted\ utterance\ counts\ in\ C_j\ that\ belong\ to\ R_i}{number\ of\ segments\ in\ R_i} \qquad (6.13)$$

Singularity score can be calculated using Equation (5.9) without any modification. Similarly, *quality score* of each sub-task is still defined as a harmonic mean of the precision, recall and singularity score.

To compute an overall quality metric over all of the sub-tasks in the reference set, a *macro-average* is used to emphasize that every sub-task is equally important. Each sub-task is assigned an equal weight when the metrics from all of the sub-tasks are averaged regardless of its size. Macro-average is discussed in detail in Section 5.4.3. The macro-average can be used to compute an overall metric of every subtask-level metric (e.g. macro-average precision and macro-average singularity score). The quality score computed from macro-average precision, macro-average recall and macro-average singularity score provides a single number that indicates the overall quality of the output clusters. This number is useful for an end-to-end comparison between two clustering algorithms.

6.2.4 Experiments and results

The test corpora consist of 24 dialogs from the air travel planning domain and 20 dialogs from the map reading domain. Both task-oriented domains are described in Section 3.2 and Section 3.4 respectively. Task structure annotation for each dialog consists of boundaries and label for each task and sub-task and a label for each concept word. The dialogs from the air travel planning domain were annotated with the task structure presented in Table 3.3 while the dialogs from the map reading domain were annotated with the task structure presented in Table 3.5. The annotation was done by a domain expert. These test corpora are similar to the ones used in dialog segmentation experiments described previously.

Table 6.35 shows the statistics of the segment types in the air travel domain. Each segment label in the table presents the full path in the task and sub-task hierarchy with the ascendant task and sub-task listed on the left. We may refer to each segment type only by its leaf node in the path (highlighted in bold) for short. The four most frequent segment types are the four sub-subtasks listed in Table 3.3. **<reserve_flight>** is a dialog segment which occurs after a **query_flight_info** sub-task and a **query_flights_fare** sub-task and discusses a ***make_a_flight_reservation*** action; please note that it does not correspond to the entire **reserve_flight** sub-task. Only these five segment types correspond to actions and forms. Other three segment types do not contain any action. **<create_an_itinerary>** is a dialog segment that is not covered by any other sub-task and corresponds to a closing sub-dialog. **<reserve_car>** and **<reserve_hotel>** are sub-dialogs while related to car and hotel reservations, do not contribute directly toward any action. For example, a client may only mention a possibility of getting a car rental without giving specific criteria of the desired car. Since the result from sub-task clustering will be used to determine different types of forms in each task-oriented domain, only the segment types that correspond to actions and forms are focused in the experiments. The infrequent segment types, which include **<create_an_itinerary>**, **<reserve_car>** and **<reserve_hotel>,** are excluded from the evaluation.

Segment type	Frequency	%
<create_an_itinerary><reserve_flight>**<query_flight_info>**	53	40.77%
<create_an_itinerary><reserve_flight>**<query_flights_fare>**	23	17.69%
<create_an_itinerary><reserve_car>**<query_car_info>**	18	13.85%
<create_an_itinerary><reserve_hotel>**<query_hotel_info>**	16	12.31%
<create_an_itinerary>**<reserve_flight>**	11	8.46%
<create_an_itinerary>	6	4.62%
<create_an_itinerary>**<reserve_car>**	2	1.54%
<create_an_itinerary>**<reserve_hotel>**	1	0.77%

Table 6.35: A list of sub-tasks in the air travel domain and their frequencies in the test corpus

Table 6.36 shows the statistic of the segment types in the map reading domain. There are only two types of dialog segments in this domain which correspond to the two sub-tasks: **draw_a_segment** and **grounding**.

Segment type	Frequency	%
<draw_a_route><draw_a_segment>	219	64.22%
<draw_a_route><draw_a_segment>**<grounding>**	122	35.78%

Table 6.36: A list of sub-tasks in the map reading domain and their frequencies in the test corpus

The bisecting *K*-means algorithm also creates an etcetera cluster by combining small clusters together to capture the sub-dialogs that are not relevant to the task. Since an etcetera cluster represents irrelevant sub-dialogs, it is excluded from the evaluation.

In all experiments, unless specify else where, the bisecting cluster re-assignment threshold (threshold-1) was set to 5%, the number of bisecting runs (B) was set to 10 and the minimum cluster size (T) was set to 5. This set of parameters is similar to the one used in the HMM-based segmentation discussed in Section 6.1.7. The maximum number of clusters (*M*) in sub-task clustering experiments is also similar to the maximum number of HMM states in the HMM-based segmentation. Since the bisecting *K*-means clustering algorithm contains a random process as the new centroids of split clusters are chosen randomly, the performance of a HMM-based segmentation is averaged from 10 runs.

6.2.4.1 Oracle sub-task clustering

The oracle clustering experiment was carried on in order to investigate the performance of the bisecting *K*-means clustering algorithm when apply to the segments that do not contain any boundary error. In oracle sub-task clustering, the input segments are reference segments manually created by a coding scheme expert. In this experiment, the reference segments also include information from concept annotation as it has been shown to be useful for dialog segmentation discussed previously. The *Label+Word* representation is used to represent concept words. Pre-defined stop words are removed from a set of features. The list of pre-defined stop words is discussed in Section 6.1.6.1.

To see the effect of the number of clusters (*M*) on clustering performance, various *M* values were used. For the air travel domain, the values of *M* range from 2 to 16 to reflect both the case when *M* is smaller than the number of the reference sub-task types and the case when *M* is larger. Table 6.37 and Figure 6.18 present the oracle clustering performance when the clustering result is evaluated against 5 reference segment types discussed earlier. The best result, the one that produces highest quality score, is obtained when *M* is equal to 10 and is underlined in Table 6.37. In addition to the evaluation metrics described in Section 6.2.3, *topic recall*, which is a ratio between the number of sub-tasks discovered by the clustering algorithm and the number of sub-task types in the reference, is also reported.

Number of clusters (M)	Topic Recall	Precision	Recall	Singularity score	Quality score
2	0.400	0.604	0.339	1.000	0.535
4	0.680	0.650	0.573	0.900	0.680
6	0.760	0.664	0.630	0.818	0.693
8	0.900	0.669	0.689	0.785	0.706
10	0.920	0.746	0.723	0.773	0.745
12	0.980	0.741	0.758	0.728	0.738
14	0.980	0.736	0.739	0.690	0.714
16	0.980	0.749	0.723	0.668	0.706

Table 6.37: Oracle clustering performance in the air travel domain

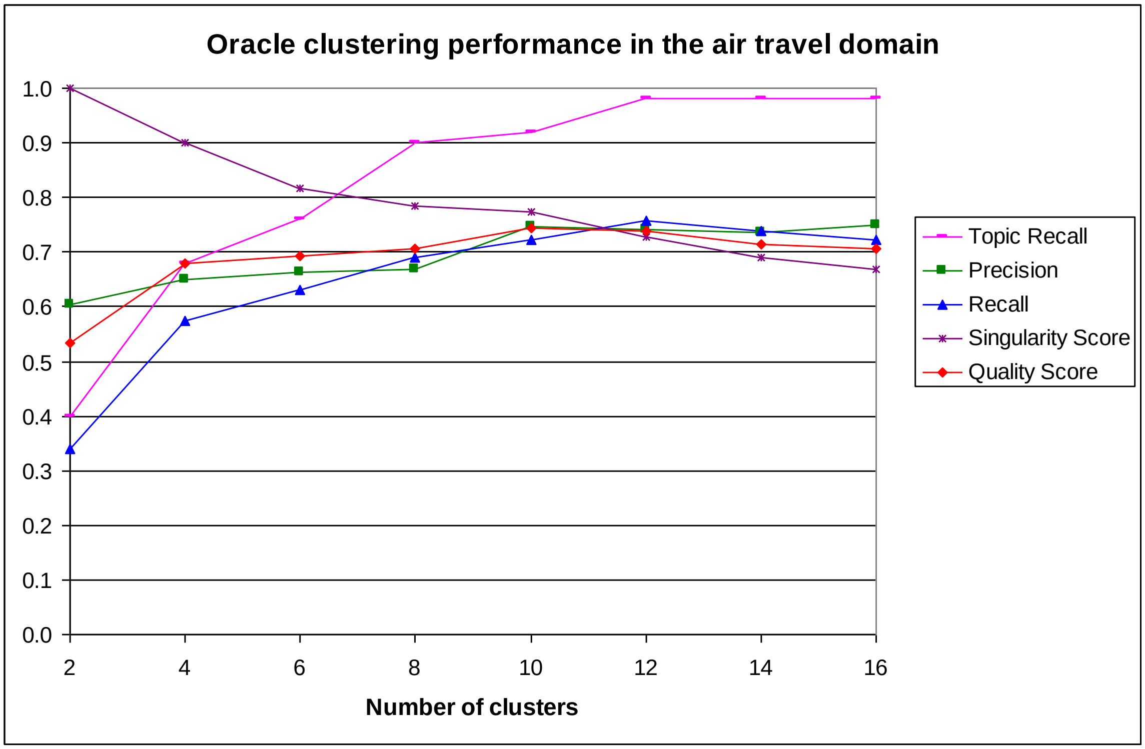


Figure 6.18: The effect of the number of clusters on clustering performance in the air travel domain

When the clustering algorithm outputs more clusters, the topic recall increases. When the value of *M* is lower than the number of sub-tasks in the reference, i.e. *M* < 5, only the frequent sub-task types can be identified. For example, when *M* = 2, only the two most frequent sub-task types, **query_flight_info** and **query_flights_fare**, are discovered. As clusters are split to produce more clusters, the less frequent sub-tasks could be discovered. However, the three infrequent segment types, **<create_an_itinerary>**, **<reserve_car>** and **<reserve_hotel>**, which do not correspond to any action and are excluded from the evaluation, are hardly identified even when *M* is set to 16. An infrequent segment type is more difficult to identify than a frequent segment type.

Sub-task	Frequency	Precision	Recall	Singularity score	Quality score
query_flight_info	53	0.937	0.908	0.275	0.513
query_flights_fare	23	0.691	0.704	0.900	0.738
query_car_info	18	0.708	0.894	0.850	0.792
query_hotel_info	16	0.843	0.644	0.944	0.800
reserve_flight	11	0.500	0.464	1.000	0.632

Table 6.38: The qualities of the sub-tasks identified by oracle clustering in the air travel domain

Table 6.38 shows the performance of the oracle clustering on each sub-task in the air travel domain when the quality score is optimal (*M* = 10). The clustering algorithm can identify a **query_flight_info** sub-task, which is the most frequent sub-task, with high precision and recall; however, the sub-task is split into 3 to 4 clusters on average for each run. Although all of the instances of a **query_flight_info** sub-task contain a similar set of concepts, the values of each concept can be different (for example, **[ArrivalCity]:**pittsburgh vs. **[ArrivalCity]:**boston and **[DepartTime]:**a.m. vs. **[DepartTime]:**p.m.). When the number of clusters increases, the clustering algorithm fails to group all of the segments that belong to a **query_flight_info** sub-task together. Those segments are split into multiple clusters; each cluster contains a similar set of concept values. For example, the segments that have the same arrival city are grouped together. For a clustering task, where the distribution of concept types is more importation than the distribution of concept values, a more abstract representation of concept words, such as the *Label* representation, may be more appropriate.

Other sub-tasks can be identified with lower precision and recall. Some instances of these sub-tasks are grouped together into the same cluster. A **query_flights_fare** sub-task, a **query_car_info** sub-task and a **query_hotel_info** sub-task share some common concepts and keywords, such as **[Fare]:**dollar and “rate”. Some of their instances also contain short closing dialogs that make them similar to a **reserve_flight** sub-task. Furthermore, these sub-tasks are shorter and have less number of occurrences than a **query_flight_info** sub-task; therefore, provide less context to the clustering algorithm. On the other hand, these sub-tasks are usually presented by a single cluster and have high singularity score.

The analysis of topic recall and the observation on the quality of each type of sub-task identified by oracle clustering reveal that the frequency of the segment type has some effect on the clustering result. A frequent segment type is likely to be identified reliably by the clustering algorithm but is subjected to splitting while an infrequent segment type may not contain enough information for the clustering algorithm to identify the similarity among its instances and thus more difficult to discover.

Number of clusters (M)	Topic Recall	Precision	Recall	Singularity score	Quality score
<u>2</u>	<u>0.950</u>	<u>0.737</u>	<u>0.730</u>	<u>0.950</u>	<u>0.791</u>
4	1.000	0.906	0.860	0.650	0.787
6	1.000	0.909	0.856	0.447	0.656
8	1.000	0.910	0.872	0.325	0.562

Table 6.39: Oracle clustering performance in the map reading domain

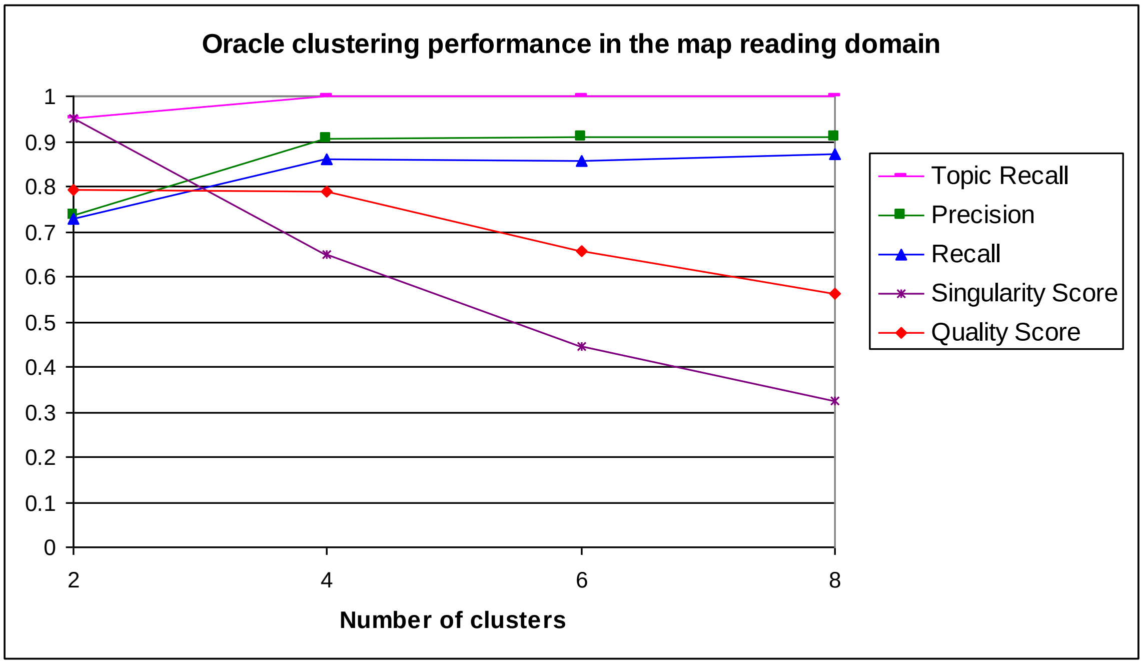


Figure 6.19: The effect of the number of clusters on clustering performance in the map reading domain

In the map reading domain, the number of clusters (*M*) was varied from 2 to 8 in the experiment. The range of *M* is smaller than the one used in the air travel domain since the number of sub-tasks is smaller. Table 6.39 and Figure 6.19 show the performance of the oracle clustering in the map reading domain. When the clustering algorithm outputs more clusters, topic recall tends to increase together with precision and recall. Singularity score, on the other hand, tends to decrease. When there is no further improvement in both precision and recall, quality score is decreased from the decrease in singularity score.

These observations are quite similar to the ones found in the air travel domain. The best result is obtained when *M* is equal to 2 and is underlined in Table 6.39.

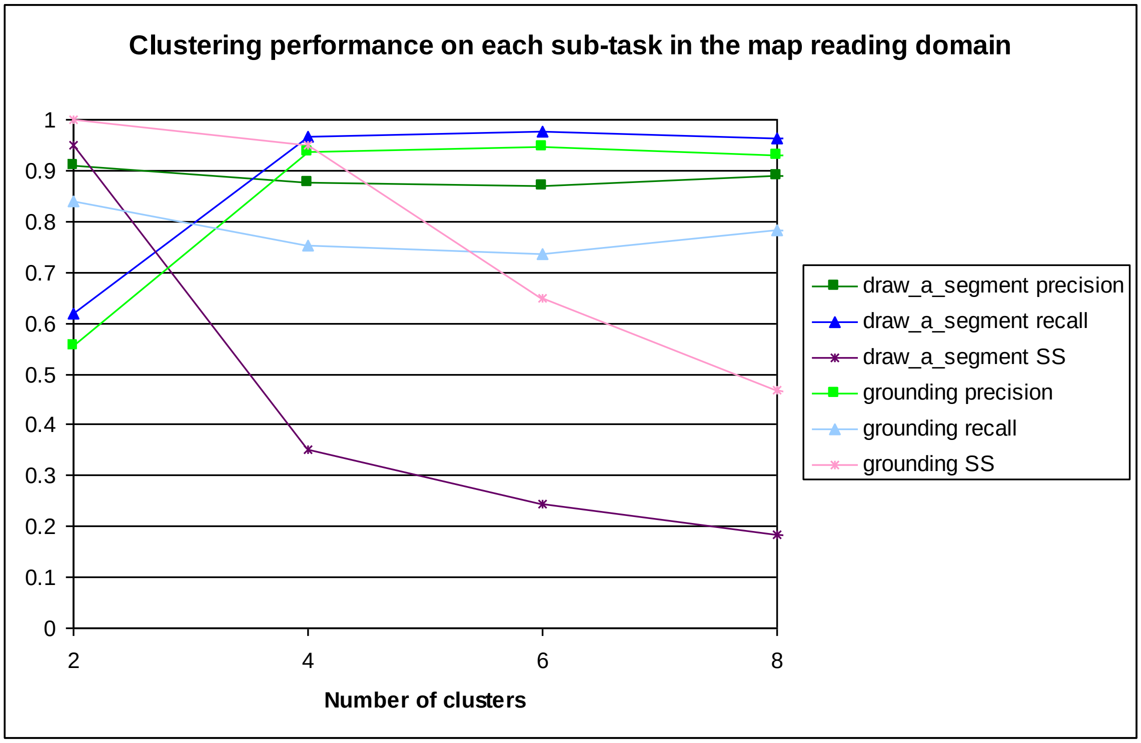


Figure 6.20: The qualities of the sub-tasks identified by oracle clustering in the map reading domain

Figure 6.20 shows the performance of the oracle clustering on each sub-task in the map reading domain. The graphs for a **draw_a_segment** sub-task are in darker colors while the graphs for a **grounding** sub-task are in lighter colors. When *M* is equal to 2 the clustering algorithm can already identify both types of sub-task; however, some occurrences of a **draw_a_segment** sub-task are merged into a **grounding** cluster and deteriorate the precision of the **grounding** sub-task and the recall of the **draw_a_segment** sub-task. Since the conversations in this domain are quite dynamic, some instances of a **grounding** sub-task may contain a partial route segment description which makes them quite similar to the instances of a **grounding** sub-task. When *M* is larger, those occurrences of a **draw_a_segment** sub-task are separated into other clusters which make both the precision of the **grounding** sub-task and the recall of the **draw_a_segment** sub-task increased. Nevertheless, the singularity score is lower for a **draw_a_segment** sub-task. Instances of a **draw_a_segment** sub-task are split into multiple clusters based on a set of concept values they contain similar to a **query_flight_info** sub-task in the air travel domain, When a cluster is split, it is likely to separate the segments that belong to different sub-tasks into different clusters; therefore, the new clusters are likely to have higher purity than the original cluster. As a result, precision usually increases. In the air travel domain, the recall also increases because the new clusters can discover new sub-tasks while in the map reading domain the new clusters produce better separation between a **draw_a_segment** sub-task and a **grounding** sub-task. When new sub-tasks are discovered the splitting does not decrease singularity score. After all of the sub-tasks are discovered (when *M* is equal to 8 in the air travel domain and when *M* is equal to 2 in the map reading domain), allowing the clustering algorithm to output slightly more number of clusters could still improve precision and recall as the sub-tasks could be better separated even though there is no new sub-task to discover. However, this comes at the expense of lower singularity score. When the number of clusters increases much further, the clustering algorithm only partitions the clusters into smaller clusters without making a better separation among the sub-tasks. Both precision and recall are not improved and quality score is lower from reduction in singularity score caused by splitting sub-tasks. Based on this observation, the optimal performance could be obtained when all of the sub-tasks in the reference are identified or when allowing the clustering algorithm to output slightly more number of clusters.

Summary

The bisecting *K*-means clustering algorithm achieves quite a good clustering result when the input segments contain no boundary error producing overall quality score of 0.745 in the air travel domain and 0.791 in the map reading domain when the number of clusters is optimal. A frequent sub-task can be identified with high precision and recall; however, it is split into multiple clusters. A more abstract concept word representation may help reduce the splitting problem. An infrequent sub-task is more difficult to discover as there is less context to identify the similarity among its instances. Precision and recall are lower; however, singularity score is quite high since it was usually presented by a single cluster.

6.2.4.2 Sub-task clustering based on predicted segments

In this experiment, the bisecting *K*-means algorithm was applied to dialog segments obtained from an automatic segmentation algorithm. These segments were produced by the TextTiling boundary predictor, TC4DTr, which utilizes information from concept annotation together with regularity weights, distance weights and a triangular smoothing scheme to determine sub-task boundaries as discussed in Section 6.1.6.6. This boundary predictor achieved good segmentation performances in both the air travel domain and the map reading domain. The TC4DTr predictor produced more dialog segments when compared to the number of segments in the reference in the air travel domain (171 vs. 130 segments) while produced fewer dialog segments when compared to the reference in the map reading domain (291 vs. 341 segments). Besides the input segments all other settings are similar to the ones used in the oracle clustering discussed above. The clustering results are shown in Table 6.40. The best result is obtained when *M* is equal to 12 and is underlined in the table.

Number of clusters (M)	**Topic Recall**	**Precision**	**Recall**	**Singularity score**	**Quality score**
2	0.300	0.471	0.253	0.750	0.403
4	0.600	0.498	0.433	0.858	0.545
6	0.720	0.493	0.475	0.785	0.552
8	0.780	0.491	0.479	0.732	0.543
10	0.920	0.480	0.484	0.759	0.545
12	0.920	0.516	0.526	0.753	0.577
14	0.900	0.518	0.521	0.643	0.551
16	0.920	0.514	0.516	0.658	0.553

Table 6.40: Sub-task clustering results in the air travel domain when predicted segments are used as an input

Since predicted segments may contain some boundary errors, the sub-task clustering result obtained from predicted segments is not as good as the clustering result obtained from reference segments. Topic recall and singularity score (SS) are slightly worse, but precision and recall are considerably lower which make the overall performance in terms of quality score lower. The performance of the clustering algorithm that uses predicted segments is compared to the performance of the clustering algorithm that uses reference segments in Figure 6.21 and Table 6.41. The quality metrics of the oracle clustering are illustrated in darker colors while the quality metrics of the predicted segment clustering are illustrated in lighter colors.

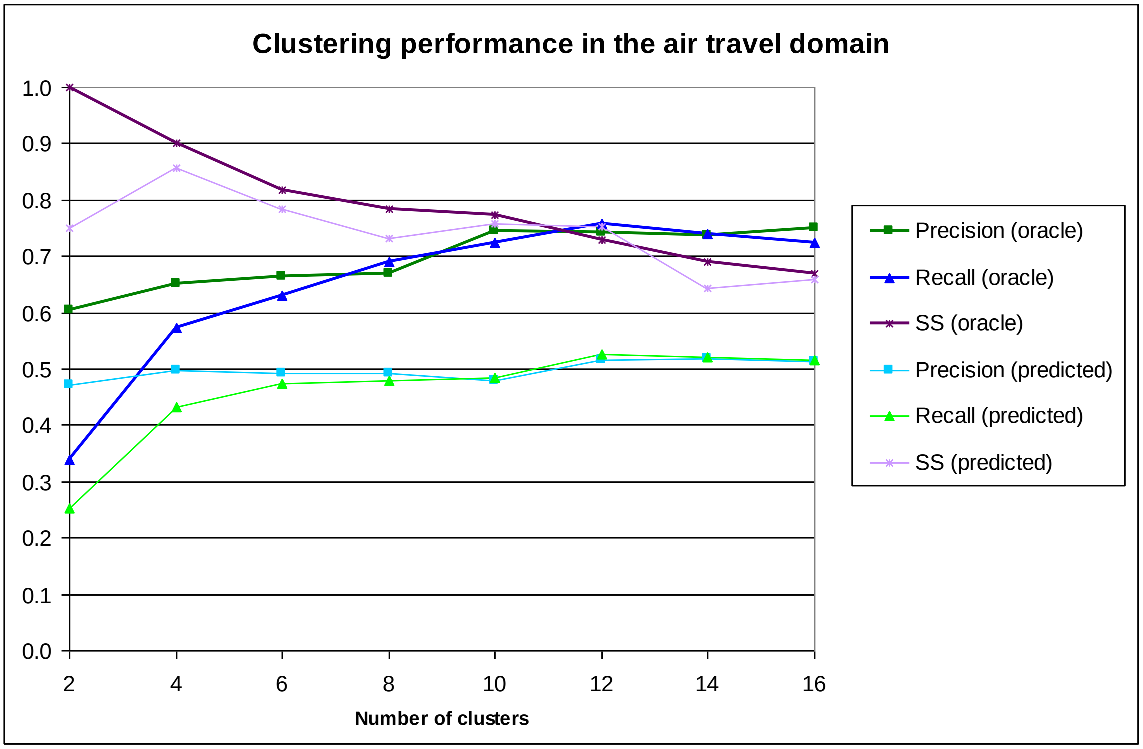


Figure 6.21: A performance comparison between a clustering algorithm that uses reference segments and the one that uses predicted segments in the air travel domain

Sub-task	Oracle clustering (M=12)				Predicted segment clustering (M=12)			
	Precision	Recall	SS	QS	Precision	Recall	SS	QS
query_flight_info	0.972	0.915	0.257	0.495	0.801	0.846	0.230	0.438
query_flights_fare	0.816	0.630	0.815	0.741	0.490	0.338	0.889	0.481
query_car_info	0.760	0.839	0.900	0.814	0.433	0.523	0.889	0.559
query_hotel_info	0.675	0.588	0.733	0.596	0.485	0.303	0.938	0.478
reserve_flight	0.494	0.818	0.950	0.690	0.348	0.619	0.900	0.513
Macro-average	**0.741**	**0.758**	**0.728**	**0.738**	**0.516**	**0.526**	**0.753**	**0.577**

Table 6.41: The quality of each sub-task identified by the oracle clustering and the predicted segment clustering in the air travel domain

Table 6.41 shows the quality of each sub-task in the air travel domain identified by both clustering algorithms. These set of results are obtained when *M* is equal to 12, which is the optimal number of clusters producing the best overall quality score for the predicted segment clustering. Please note that the quality score of the oracle clustering at *M* equals 12 is slightly lower than the optimal value obtained when *M* equals 10. The quality of a **query_flight_info** sub-task identified from predicted segments is almost as good as the one identified from reference segments. Precision is slightly lower mostly because consecutive utterances that belong to a **query_flights_fare** sub-task are also included in the same cluster when the boundaries between a **query_flight_info** sub-task and a **query_flights_fare** are missing. For other sub-tasks, especially **query_flights_fare**, **query_car_info** and **query_hotel_info**, both precision and recall are considerably lower. Many boundaries of these sub-tasks are missing as shown in Table 6.19. Even with correct sub-task boundaries as in the oracle clustering, some instances of these sub-tasks still get merged into the same cluster. When the boundaries among them are missing, it is more difficult to identify the sub-tasks accurately.

Nevertheless, some types of boundary errors have less negative effect toward the clustering performance. A missing boundary between two instances of the same sub-task type is less problematic if the merged segment could be assigned to the right cluster. The result shows that, although, many boundaries between consecutive instances of a **query_flight_info** sub-task are not identified, they do not have much effect on the clustering quality of a **query_flight_info** sub-task. Extra boundaries, or false alarms, are less problematic than missing boundaries since the fragments of the same sub-task could be grouped together into the same cluster. A **query_flight_info** sub-task which has more false alarms than other types of sub-tasks still achieves better clustering quality. However, too many false alarms could affect the clustering performance since small fragments do not contain enough context to accurately identify the similarity among the fragments as occurred with utterance-based segmentation discussed in Section 6.1.7.1.

The sub-task clustering performance in the map reading domain obtained when the predicted segments are used is shown in Table 6.42 and a comparison between the predicted segment clustering and the oracle clustering is given in Figure 6.22 and Table 6.43. The quality metrics of the oracle clustering are illustrated with darker colors while the quality metrics of the predicted clustering are illustrated with lighter colors. Both precision and recall are considerably lower than when the reference segments are used regardless of the number of clusters outputted. However, when M is large, the predicted segment clustering obtains higher singularity score which reduces the difference between the quality scores of both algorithms. Better overall singularity score comes from higher singularity score of a grounding sub-task. Nevertheless, the optimal quality score produced by the predicted segment clustering, which is obtained when M is equal to 4 and is underlined in Table 6.42, is lower than the optimal quality score produced by the oracle clustering (QS = 0.791 when M is equal to 2).

Number of clusters (M)	Topic Recall	Precision	Recall	Singularity score	Quality score
2	0.500	0.642	0.500	0.500	0.540
4	1.000	0.701	0.651	0.675	0.675
6	1.000	0.714	0.645	0.585	0.641
8	1.000	0.724	0.637	0.502	0.597

Table 6.42: Sub-task clustering results in the map reading domain when predicted segments are used as an input

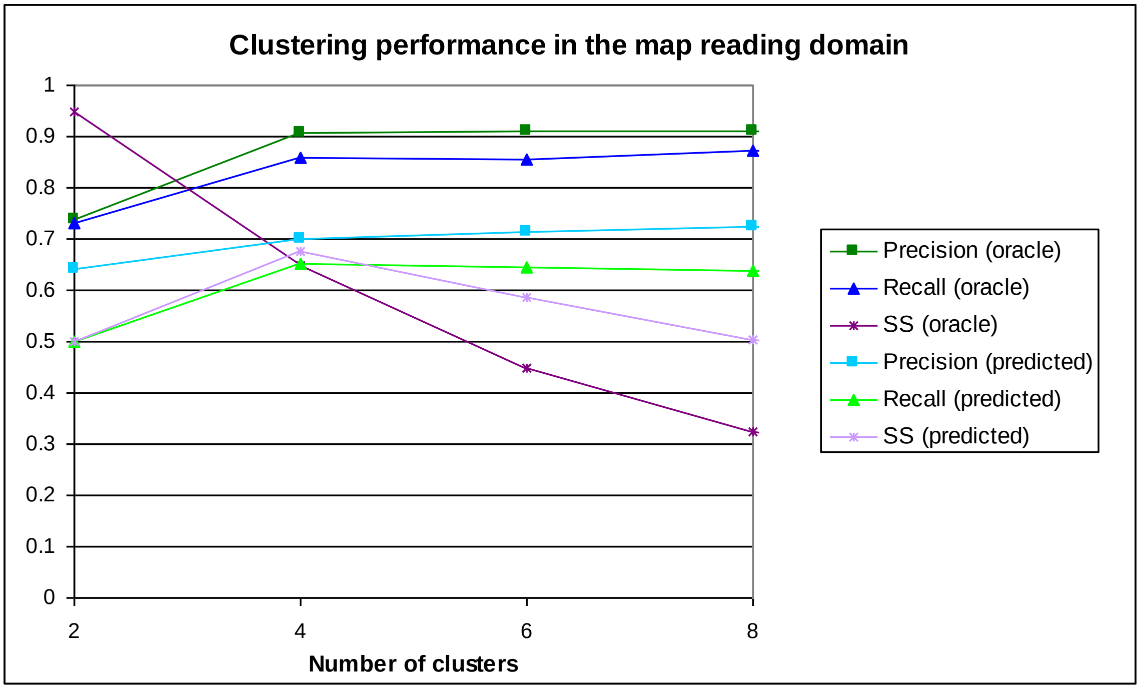


Figure 6.22: A performance comparison between a clustering algorithm that uses reference segments and the one that uses predicted segments in the map reading domain

Sub-task	Oracle clustering (M=4)				Predicted segment clustering (M=4)			
	Precision	Recall	SS	QS	Precision	Recall	SS	QS
draw_a_segment	0.876	0.968	0.350	0.593	0.737	0.854	0.350	0.553
grounding	0.935	0.752	0.950	0.862	0.665	0.449	1.000	0.620
Macro-average	**0.906**	**0.860**	**0.650**	**0.787**	**0.701**	**0.651**	**0.675**	**0.675**

Table 6.43: The quality of each sub-task identified by the oracle clustering and the predicted segment clustering in the map reading domain

The quality metrics of each sub-task in the map reading domain presented in Table 6.43 are obtained when *M* is equal to 4, which is the optimal number of clusters for the predicted segment clustering. Please note that the overall quality score of the oracle clustering at *M* equals 4 is slightly lower than its optimal value. The quality of a **draw_a_segment** sub-task identified from predicted segments is almost as good as the one identified from reference segments. The precision is slightly lower mostly because some consecutive utterances that belong to a **grounding** sub-task are also included in the same cluster when the sub-task boundary predictor failed to identify a boundary between a **draw_a_segment** sub-task and a **grounding** sub-task. This problem also causes the recall of a **grounding** sub-task to be much lower. On the other hand, missing boundaries between consecutive instances of a **draw_a_segment** sub-task and false alarm boundaries have less negative effect on the clustering results.

Summary

The quality of the clusters obtained from predicted segments is lower than the quality of the clusters obtained from the reference segments as inaccurate segment boundaries affect the performance of the clustering algorithm. However, some types of boundary errors have more negative influence on the clustering performance than other types of errors. A missing boundary between different types of sub-tasks is usually more problematic than a missing boundary between two consecutive instances of the same type of sub-task. False alarm boundaries also have less negative effect on the clustering result.

Although, some of the predicted segments have inaccurate segment boundaries, the quality of some sub-tasks, such as a **query_flight_info** sub-task and a **draw_a_segment** sub-task, identified from the predicted segments is almost as good as the ones identified from the reference segments.

6.2.4.3 The label representation

In this experiment, the *Label* representation, which focuses on a concept label rather than its value as it is more informative for sub-task clustering, is used to represent concept words. As discussed in Section 6.2.4.1, a more abstract concept word representation could also reduce the splitting problem that usually occurs with high frequency sub-tasks. A performance comparison between the two concept word representations, the one that uses both a concept label and a word string (the *Label+Word* representation) and the one that uses only a concept label (the *Label* representation) is given in Table 6.44. For both representations, the results are the optimal result obtained from the oracle clustering. For the *Label+Word* representation the optimal result is obtained when M is equal to 10 and is similar to the one presented in Table 6.38 while for the *Label* representation the optimal result is obtained when *M* is equal to 14.

Sub-task	**Label+Word (M=10)**				**Label (M=14)**			
	Precision	**Recall**	**SS**	**QS**	**Precision**	**Recall**	**SS**	**QS**
query_flight_info	0.937	0.908	0.275	0.513	1.000	0.906	0.242	0.475
query_flights_fare	0.691	0.704	0.900	0.738	0.930	0.848	0.950	0.897
query_car_info	0.708	0.894	0.850	0.792	0.835	0.706	0.850	0.760
query_hotel_info	0.843	0.644	0.944	0.800	0.775	0.738	0.517	0.630
reserve_flight	0.500	0.464	1.000	0.632	0.523	0.909	0.950	0.724
Macro-average	**0.746**	**0.723**	**0.773**	**0.745**	**0.812**	**0.821**	**0.702**	**0.772**

Table 6.44: Clustering results for each sub-task in the air travel domain when different concept word representations are used by the oracle clustering

The optimal quality score for the *Label* representation is 0.772 which is slightly better than the value of 0.745 achieved by the *Label+Word* representation. The *Label* representation achieves higher precision and recall than the *Label+Word* representation. However, the singularity score is lower because its optimal result contains more clusters. When compare singularity scores of both representations at the same *M* value, the *Label* representation has a slightly better singularity score. The quality of each sub-task is also given in Table 6.44. The *Label* representation achieves better quality score for a **query_flights_fare** sub-task and a **reserve_flight** sub-task due to higher precision and recall. For a **query_flight_info** sub-task and a **query_hotel_info** sub-task, it achieves lower quality score mainly from more splitting as the number of clusters is higher. The quality score of a **query_car_info** sub-task is also slightly lower.

The *Label* representation does not improve singularity score as expected. Frequent sub-tasks still get split into multiple clusters. Although, dialog segments that belong to the same sub-task should contain a similar set of concepts, some concepts are optional and do not occur in every instances of the sub-task. For instance, **ArrivalAirport** and **ArrivalState** are optional concepts and do not occur in every **query_flight_info** sub-task. A frequent sub-task may get split into multiple clusters based on the optional concepts and other non-concept words that they contain as the bisecting *K*-means clustering algorithm always splits the largest cluster first.

All splitting problems discussed above are the case that a sub-task is represented by more than one cluster which occurs when the instances of the same sub-task are separated into multiple clusters and those instances are also a majority sub-task in some of the clusters. This type of splitting is reflected by singularity score. However, if the instances of the same sub-task are assigned to multiple clusters but as a minority sub-task in some of those clusters, there might be only one cluster that represents the sub-task. Singularity score cannot capture this kind of splitting; nevertheless, the splitting causes the clusters to be heterogeneous and reduces precision. Recall also decreases as some dialog segments are assigned to the clusters that do not represent their corresponding sub-task.

The *Label* representation, which is a more abstract representation, makes it easier to distinguish some sub-tasks from other sub-tasks if they contain different sets of concepts and alleviates the second splitting problem. Precision and/or recall are increased for those sub-tasks. A **query_flight_info** sub-task, a **query_flights_fare** sub-task and a **reserve_flight** sub-task can be identified with high recall since most of their instances contain a similar set of concepts. In terms of precision, a **query_flight_info** sub-task has a perfect precision (i.e. precision = 1) as it contains a set of concepts that is distinct from the ones used in other sub-tasks. A **query_flights_fare** sub-task also has high precision. However, precision of a **reserve_flight** sub-task is quite low since the cluster that represents this sub-task also includes dialog segments from other sub-tasks that contain similar closing sub-dialogs. A **query_hotel_info** sub-task and a **query_car_info** sub-task have lower recall than other three sub-tasks since some of their instances get merged into the clusters that represent other sub-tasks. For instance, some instances of a **query_hotel_info** sub-task and a **query_car_info** are assigned to the cluster that represents a **query_flights_fare** sub-task as all three sub-tasks share the same concept **Fare**.

Sub-task	**Label+Word (M=2)**				**Label (M=2)**			
	Precision	**Recall**	**SS**	**QS**	**Precision**	**Recall**	**SS**	**QS**
draw_a_segment	0.911	0.620	0.950	0.775	0.997	0.991	1.000	0.996
grounding	0.555	0.840	1.000	0.773	0.984	0.995	1.000	0.993
Macro-average	**0.737**	**0.730**	**0.950**	**0.791**	**0.991**	**0.993**	**1.000**	**0.994**

Table 6.45: Clustering results for each sub-task in the map reading domain when different concept word representations are used by the oracle clustering

Table 6.45 shows a performance comparison between the two concept word representations in the map reading domain. For both representations, the results are the optimal result achieved by the oracle clustering which are obtained when *M* is equal to 2. When the *Label* representation is used all of the quality metrics, precision, recall and singularity score, are almost perfect for both sub-tasks. The *Label* representation, which focuses on the concepts themselves rather than their values, can perfectly distinguish between a **draw_a_segment** sub-task and a **grounding** sub-task since both sub-tasks contain disjoint sets of concepts. There is no splitting in the optimal result obtained from the *Label* representation since the optimal number of clusters is equal to the number of the reference sub-tasks. However, when the clustering algorithm outputs more clusters, both sub-tasks are split into many clusters which reduces the singularity score. Nevertheless, precision and recall for both sub-tasks are still close to perfect.

Table 6.46 shows a performance comparison between the two concept word representations when predicted segments from the air travel domain are used as an input of a clustering algorithm. The results are the optimal result obtained when *M* is equal to 12 for the *Label+Word* representation and when *M* is equal to 10 for the *Label* representation.

Sub-task	Label+Word (M=12)				Label (M=10)			
	Precision	Recall	SS	QS	Precision	Recall	SS	QS
query_flight_info	0.801	0.846	0.230	0.438	0.865	0.844	0.213	0.424
query_flights_fare	0.490	0.338	0.889	0.481	0.464	0.629	0.950	0.619
query_car_info	0.433	0.523	0.889	0.559	0.717	0.049	1.000	0.285
query_hotel_info	0.485	0.303	0.938	0.478	0.416	0.508	1.000	0.578
reserve_flight	0.348	0.619	0.900	0.513	0.377	0.735	0.733	0.529
Macro-average	**0.516**	**0.526**	**0.753**	**0.577**	**0.549**	**0.553**	**0.741**	**0.601**

Table 6.46: Clustering results for each sub-task in the air travel domain when different concept word representations are used by the predicted segment clustering

The *Label* representation achieves slightly better quality score from higher precision and recall; however, singularity score is slightly lower. When consider the quality of each sub-task separately, the *Label* representation achieves better quality score for a **query_flights_fare** sub-task and a **query_hotel_info** sub-task mainly from higher recall. On the other hand, a **query_car_info** sub-task has much lower quality score due to much lower recall. The recall is lower because many dialog segments that belong to a **query_car_info** sub-task get merged into either a **query_flights_fare** sub-task or a **query_hotel_info** sub-task as many boundaries between a **query_car_info** sub-task and a **query_flights_fare** sub-task or a **query_hotel_info** sub-task are missing as shown in Table 6.19. Moreover, some instances of these three sub-tasks are quite similar since they share some common concepts and keywords, such as **[Fare]** and "rate" as discussed earlier.

Sub-task	**Label+Word (M=4)**				**Label (M=2)**			
	Precision	**Recall**	**SS**	**QS**	**Precision**	**Recall**	**SS**	**QS**
draw_a_segment	0.737	0.854	0.350	0.553	0.816	0.846	1.000	0.881
grounding	0.665	0.449	1.000	0.620	0.705	0.657	1.000	0.761
Macro-average	**0.701**	**0.651**	**0.675**	**0.675**	**0.760**	**0.752**	**1.000**	**0.823**

Table 6.47: Clustering results for each sub-task in the map reading domain when different concept word representations are used by the predicted segment clustering

Table 6.47 shows a performance comparison between two clustering algorithms that use predicted segments from the map reading domain as an input but use different concept word representations. The results are the optimal results obtained when *M* is equal to 4 for the *Label+Word* representation and when *M* is equal to 2 for the *Label* representation. The algorithm that uses the *Label* representation outperforms the algorithm that uses the *Label+Word* representation both in terms of the overall performance (the macro-average) and the performance on each sub-task. The best performance achieves by the predicted segment clustering that uses the *Label* representation is also better than the best performance achieves by the oracle clustering that uses the *Label+Word* representation shown Table 6.39. All of the quality metrics, precision, recall and singularity score, are slightly better. This result confirms that concept labels are more informative than their values for sub-task clustering and that the *Label* representation is a better representation than the *Label+Word* representation. An appropriate feature representation provides more useful information to the clustering algorithm than accurate segment boundaries.

Summary

The *Label* representation, which uses only a concept label to represent a concept word, is a better representation than the *Label+Word* representation, which uses both a concept label and a word string to represent a concept word, for sub-task clustering. The *Label* representation can better distinguish among different types of sub-tasks if they contain different sets of concepts. If the sets of concepts used in all of the reference sub-tasks are disjoint, a clustering algorithm that uses the *Label* representation can achieves a very good results as in the map reading domain. The performance is almost perfect when the reference segments are used as an input. The quality of the clusters is still good when the predicted segments are used. Substantial improvement is achieved when compared to the quality of the clusters obtained from the algorithm that uses the *Label+Word* representation. The result of the clustering algorithm that uses the *Label* representation with predicted segments is even better than the result obtained from the oracle clustering that uses the *Label+Word* representation. This result confirms that the *Label* representation is more suitable for sub-task clustering than the *Label+Word* representation. It also demonstrates that an appropriate feature representation provides more useful information to the clustering algorithm than accurate segment boundaries.

In the air travel domain, the performance gain from the *Label* representation is smaller since the sub-tasks contain overlapping sets of concepts. A sub-task can be identified with high recall if most of its instances contain a similar set of concepts, such as, a **query_flights_fare** sub-task and a **reserve_flight** sub-task. High precision can be achieved if a sub-task contains a set of concepts that is distinct from the ones used in other sub-tasks such as a **query_flight_info** sub-task. For the oracle clustering, the new concept word representation improves the overall precision in recall. However, frequent sub-tasks still get split into multiple clusters based on the optional concepts and other non-concept words that they contain as the bisecting *K*-means clustering algorithm always splits the largest cluster first. As a result, the *Label* representation improves singularity score just slightly.

In addition to higher overall cluster quality, in all of the experiments, the variances of the quality metrics of all 10 runs are also lower when the *Label* representation is used.

6.2.4.4 Slot extraction

In this experiment, the best clustering result obtained when the predicted segments are used as an input of the clustering algorithm is analyzed to identify a set of slots that is associated with each type of form. This result is obtained when the *Label* representation is used is discussed in detail in the previous section. Table 6.48 shows a list of slots and their frequencies of each cluster in the air travel domain. For simplicity, only the frequent slots are presented.

Sets of slots for a flight query form, a hotel query form and a fare query form extracted from the corresponding clusters are quite acceptable. There are 5 clusters that represent a **query_flight_info** sub-task. All frequent slots found in these clusters belong to a flight query form. Erroneous slots that belong to other types of forms occur only a few times and are not shown in Table 6.48. A query_flight_info(3) cluster and a query_flight_info(4) cluster contain most essential slot in a flight query form. However, other clusters miss some of the necessary slots or contain only a few instances of them since false alarm boundaries may cause some instances of a **query_flight_info** sub-task to be separated into multiple segments. For example, a query_flight_info(1) cluster misses the departure date related slots. A cluster that corresponds to a **query_hotel_info** sub-task contains all of the necessary slots in a hotel query form. It also contains only a few instances of erroneous slots. A cluster that corresponds to a **query_flights_fare** sub-task contains some erroneous slots that belong to a hotel query form and a car query form as some instances of these forms get merged into the same cluster as discussed earlier. Erroneous slots that belong to other sub-tasks are highlighted in red.

However, since the recall of a **query_car_info** sub-task is very low as many of its instances get merged into either a **query_flights_fare** sub-task or a **query_hotel_info** sub-task, many essential slots, such as **[car_rental_company]** and **[car_category]**, are missing. In the reference, a **reserve_flight** sub-task does not contain any concept as it only discusses a ***make_a_flight_reservation*** action at the end of a dialog when all other information has been discussed. The extraneous slots come from other types of sub-tasks that get merged into the cluster as its precision is quite low. The quality of each sub-task is presented in Table 6.46.

query_flight_info (1)		**query_flight_info (2)**		**query_flight_info (3)**		**query_flight_info (4)**	
[arrive_time:minute_number]	63	[airline_company]	79	[airline_company]	39	[depart_time:minute_number]	65
[arrive_time:hour_number]	42	[arrive_time:minute_number]	46	[arrive_loc:city]	17	[depart_time:hour_number]	60
[depart_time:minute_number]	37	[depart_time:hour_number]	40	[arrive_time:minute_number]	15	[arrive_loc:city]	60
[depart_time:hour_number]	36	[depart_time:minute_number]	39	[fare]	14	[arrive_time:minute_number]	52
[depart_time:am_pm]	31	[arrive_time:hour_number]	36	[depart_time:time_period]	12	[depart_time:am_pm]	46
[arrive_loc:city]	31	[arrive_loc:city]	27	[depart_loc:city]	10	[depart_loc:city]	42
[depart_loc:city]	26	[depart_time:am_pm]	23	[arrive_time:hour_number]	10	[arrive_time:hour_number]	35
[arrive_loc:airport]	23	[flight_number]	15	[depart_date:month_number]	10	[depart_date:month_number]	31
[airline_company]	19	[arrive_time:am_pm]	14	[depart_time:minute_number]	9	[airline_company]	30
[arrive_time:am_pm]	18	[arrive_loc:airport]	13	[depart_time:hour_number]	8	[flight_ref]	26
		[depart_loc:city]	13				
		[depart_time:time_period]	11				
query_flight_info (5)		**reserve_flight (1)**		**reserve_flight (2)**		**query_flights_fare**	
[depart_date:month_number]	43	[area]	10	[city]	4	[fare]	257
[depart_date:month]	26	[arrive_loc:city]	6	[car_category]	3	[city]	27
[arrive_loc:city]	24	[arrive_loc:airport]	5	[depart_date:day_of_week]	2	[car_rental_company]	17
[depart_loc:city]	12	[arrive_date:day_of_week]	4	[arrive_loc:city]	2	[hotel_name]	15
[depart_date:year]	12	[airline_company]	3	[hotel_name]	2	[arrive_loc:city]	14
[depart_date:day_of_week]	10	[depart_loc:city]	3	[arrive_time:hour_number]	2	[airline_company]	11
[depart_time:time_period]	10	[fare]	3	[flight_ref:airline_company]	2		
query_hotel_info		**query_car_info**					
[fare]	75	[car_type]	13				
[city]	36	[city]	3				
[hotel_name]	33	[state]	1				
[area]	28						
[arrive_date:month_number]	14						
[depart_date:month_number]	10						

Table 6.48: A list of slots and their frequencies from each cluster in the air travel domain.

draw_a_segment		**grounding**	
[Orientation:direction]	728	[TargetLM:landmark]	309
[Path:Location:landmark]	596	[Location:location_mod]	288
[EndLocation:Location:landmark]	433	[Location:landmark]	207
[Path:Location:location_mod]	416	[Location]	192
[Path:Location]	388	[Orientation:direction]	93
[EndLocation:Location:location_mod]	385	[Location:map_ref]	93
[Orientation:direction_mod]	315	[EndLocation:Location:landmark]	59
[EndLocation:Location]	290		
[EndLocation:Location:map_ref]	179		
[StartLocation:Location:landmark]	124		
[TargetLM:landmark]	119		
[Distance:amount]	96		

Table 6.49: A list of slots and their frequencies from each cluster in the map reading domain.

Table 6.49 shows a list of slots and their frequencies of each cluster in the air travel domain. A list of slots extracted from a draw_a_segment cluster and a list of slots extracted from a grounding cluster contain all of the necessary slots for a segment description form and a grounding form respectively. Since both clusters are not homogenous, they also contain some erroneous slots. Nevertheless, those erroneous slots, highlighted in red, do not occur as frequent as most of the correct slots that belong to the forms.

Summary

Essential slots in most of the forms can be identified from clusters of predicted segments in both the air travel domain and the map reading domain. If the recall of the sub-task is not too low, all of the essential slots of the corresponding form can be identified as only multiple instances of a concept, not all of them, is sufficient to identify the corresponding slot. If a cluster is not homogenous, i.e. precision is not perfect, it may contain extraneous slots that belong to other form types. However, if the precision is not too low these extraneous slots will occur only a few times compared to the correct ones. A **query_car_info** sub-task has very low recall, therefore, some of the slots can not be identified. For a **reserve_flight** sub-task, the corresponding clusters contain extraneous slots that are quite frequent since the precision is quite low.

6.2.5 Discussion and conclusion

In the previous sections, an unsupervised clustering algorithm, the bisecting *K*-means clustering algorithm, has been applied to the problem of sub-task clustering through a series of experiments. In this section, I will summarize the clustering results and interesting findings from those experiments.

The quality of the clusters obtained from predicted segments is lower than the quality of the clusters obtained from the reference segments as inaccurate segment boundaries affected the performance of the clustering algorithm. However, some types of boundary errors have more negative influence on the clustering performance than other types of errors. A missing boundary between different types of sub-tasks is usually more problematic than a missing boundary between two consecutive instances of the same type of sub-task. False alarm boundaries also have less negative effect on the clustering result. Although, some of the predicted segments have inaccurate segment boundaries, the quality of some sub-tasks that occur frequently, such as a **query_flight_info** sub-task and a **draw_a_segment** sub-task, identified from the predicted segments is almost as good as the ones identified from the reference segments.

The frequency of the segment type has some effect on the clustering result. A frequent segment type is likely to be identified reliably by the clustering algorithm but is subjected to splitting as the bisecting *K*-means clustering algorithm always splits the largest cluster first while an infrequent segment type may not contain enough information for the clustering algorithm to identify the similarity among its instances and thus more difficult to discover.

Since a list of concepts that occurs in one sub-task is distinguishable from a list of concepts that occurs in other sub-tasks regardless of the values of the concepts, concept labels are more informative for sub-task clustering than concept values. The *Label* representation, which focuses only on a concept label not its values, is a better representation than the *Label+Word* representation, which uses both a concept label and a word string to represent a concept word. The clustering algorithm that uses the *Label* representation achieves a better clustering performance in both domains. When the sets of concepts used in all of the sub-tasks are disjoint, a clustering algorithm that uses the *Label* representation can achieves a very good results as in the map reading domain. With a more abstract concept word representation, the quality of the clusters obtained from predicted segments is even better than the quality of the clusters obtained from the reference segments that use the *Label+Word* representation. This result demonstrates that an appropriate feature representation provides more useful information to the clustering algorithm than accurate segment boundaries. However, when the sub-tasks contain overlapping sets of concepts as in the air travel domain, the performance gain obtained from the *Label* representation is small. In addition to higher overall cluster quality, in all of the experiments, the variances of the quality metrics of all 10 runs are also lower when the *Label* representation is used.

For slot extraction, A set of slots in each form type can be identified by extracting a list of concepts from the corresponding clustering. When the best clustering result obtained from predicted segments are analyzed, essential slots for most of the forms can be identified in both domains. Only moderate sub-task precision and recall are required in order to accurately identify a set of slots. However, if the recall is too low, some slots might be missing. On the other hand, if the precision is too low, the cluster might contain extraneous slots from other form types that are as frequent as the correct slots.

Sub-task clustering still has room for improvement. Since inaccurate segment boundaries affected the performance of the clustering algorithm, a better dialog segmentation algorithm will not only improve the quality of the segments but also the quality of a sub-task clustering results. To reduce the number of splitting in frequent sub-tasks, different criteria for choosing a cluster to split at each iteration can be experimented. For example, a criterion based on the overall similarity of a cluster could be used instead of the one that always chooses to split the largest cluster.

Chapter 7

Conclusion

7.1 Summary of results

In this dissertation, I proposed a *form-based dialog structure representation* (a three-level structure of *task*, sub-*task*, and *concept*) suitable for representing the domain-specific information required to build a task-oriented system and demonstrated that this representation has all of the required properties.

- *Sufficiency*

  The form-based dialog structure representation can account for 93% of dialog content in four task-oriented domains (air travel planning, bus schedule enquiry, map reading and UAV flight simulation) as measured by a *dialog coverage* discussed Section 4.1. Some limitations of the form-based representation are discussed in Section 3.8.

- *Generality*

  The form-based dialog structure representation can be applied to five disparate task-oriented domains, including air travel planning (information-accessing), bus schedule inquiry (information-accessing), map reading (problem-solving), UAV flight simulation (command-and-control), and meeting, with an exception of the tutoring domain.

- *Learnability*

  The form-based dialog structure representation is learnable by non-expert annotators and by unsupervised machine learning algorithms. The form-based representation can be applied reliably by non-expert annotators, producing high *acceptability* on task structure designs (bracketed precision > 0.8) in two disparate domains: air travel planning (an information-access task) and map reading (a problem-solving task). To show this, I introduced a novel evaluation procedure called *cross-annotator correction* suitable for comparing different markup schemes. High annotation scheme reliability suggests that the annotation scheme is concrete and unambiguous which implies learnability. Components of the form-based representation can be identified with acceptable accuracy through unsupervised machine learning approaches as summarized below.

The results of both dialog structure acquisition problems, concept identification and form identification, show that it is feasible to acquire the domain-specific knowledge necessary for creating a task-oriented dialog system automatically from a corpus of in-domain conversations using unsupervised learning approaches. With some modifications, the proposed unsupervised learning approaches are able to learn the structure of a spoken dialog which captures the required domain-specific information as represented by the form-based dialog structure representation.

Domain concepts can be identified with acceptable accuracy. The best concept identification result is obtained from the Kullback-Liebler-based clustering algorithm that uses an average linkage distance measure. The quality score of 0.70 is achieved when the automatic stopping criterion is applied. The clustering result has high precision but moderate recall since the statistical clustering algorithm cannot accurately identify infrequent concept words due to a sparse data problem. For most statistical clustering algorithms, we are able to identify automatic stopping criteria that yield close to optimal results.

Forms that occur frequently can be identified with moderate accuracy. Forms and their associated slots are identified in three sequential steps: dialog segmentation (sub-task boundary prediction), sub-task clustering, and slot extraction. To handle fine-grained segments in spoken dialogs, TextTiling and HMM-based segmentation algorithms are augmented with a data-driven stop word list and distance weights. With the proposed modifications, significant improvement on sub-task boundary prediction is achieved. Subsequent steps of form identification are subjected to propagated errors from its preceding steps. Since the proposed learning algorithms are based on generalization of recurring patterns, they can still learn from inaccurate information given that the number of errors is moderate, so that there are enough correct examples to learn from. The results show that moderate segmentation accuracy is sufficient for identifying frequent form types using the bisecting *K*-mean sub-task clustering algorithm. Similarly, moderate sub-task clustering accuracy is sufficient for identifying essential slots in each form. Dialog structure acquisition, as in the case of this thesis, does not require high learning accuracy.

In conclusion, to represent a dialog for a learning purpose (to acquire domain knowledge necessary for creating a task-oriented dialog system) we based our representation, a form-based dialog structure representation, on an observable structure. This observable representation can be generalized for various types of task-oriented dialogs and can be understood and applied by different annotators. More importantly, the representation can be learned by unsupervised learning approaches.

7.2 Contributions

This dissertation has presented the work on exploring the feasibility of using an unsupervised machine learning approach to infer the domain-specific information required to build a task-oriented dialog system through the observation of in-domain human-human conversations. The main contributions of this work are:

1. A dialog structure framework based on a form representation that has the following properties:
   - *Sufficiency*

     The proposed form-based dialog structure representation formally defines the domain-specific knowledge necessary for building a task-oriented dialog system based on the notion of form. These components while already exist, no formal definition has been specified.
   - *Generality*

     The definitions of domain knowledge components provided in the proposed form-based dialog structure representation are extended beyond the common interpretations used specifically for an information-accessing domain; hence, they are domain-independent and can be applied to different types of task-oriented dialogs as demonstrated in Chapter 3.
   - *Learnability*

     To represent a dialog for a learning purpose (acquiring domain knowledge necessary for creating a task-oriented dialog system), we based our representation on an observable structure of a dialog. As a result, this observable representation can be learned through unsupervised learning approaches.
   - A concrete mapping between dialog structure components and dialog system architecture
2. An unsupervised machine learning approach for inferring domain information from in-domain conversations that could potentially reduce human effort in developing a new task-oriented dialog system

- An unsupervised algorithm that can identify and cluster domain concepts from un-annotated data
- Automatic stop criteria for the Kullback-Liebler-based clustering algorithm and the mutual information-based clustering algorithm that yield close to optimal clustering results
- A data-driven approach that identifies less informative words in a text segmentation problem from word distribution
- An unsupervised segmentation algorithm that can identify the boundaries between fine-grained segments

3. Annotation assessment based on *cross-annotator correction* suitable for assessing inter-annotator agreement when annotators create and use their own markup schemes and annotation variations are acceptable

7.3 Future directions

Automatic domain knowledge acquisition intended for creating a new dialog system is a very new research area. This thesis has investigated the feasibility of such approach which could be extended in many directions as summarized below.

7.3.1 Extending the form-based dialog structure representation

The form-based dialog structure representation only models the observable structure of a dialog using a simple representation so that the domain-specific information captured by this representation could be inferred from in-domain conversations through existing unsupervised learning approaches. This representation works well for many types of task-oriented domains as shown in Chapter 3. The form-based representation should also be able to represent other types of task-oriented dialogs that have the following characteristics: 1) the conversation goal is achieved through the execution of domain actions, and 2) the dialog participants have to communicate the information required to perform these actions through dialog.

By assuming that all the required domain-specific information is observable and can be represented by a simple model, the form-based representation is not suitable for modeling a complex dialog that has a dynamic structure. These types of dialogs, for example, the tutoring domain, require a more expressive representation. Nevertheless, I believe that the components in the form-based dialog structure representation (i.e. task, sub-task, and concept) are still the keys components of a task-oriented dialog. However, the relations among these components in some domains may be more complex than the ones modeled by the form-based representation. For example, a compositional structure of a dialog may better be described by the purposes of the segments (Grosz and Sidner, 1986) than by the characteristics of a task and domain actions[1]. In each of the segments, some pieces of domain information are exchanged by the participants, but these pieces of information can be more complicated than domain entities modeled by concepts.

Some modifications can be applied to the form-based dialog structure representation to make it accounted for more types of dialogs. Concepts could be extended to model other kind of domain information not just domain entities, for example, relations between entities such as equality. For a complex domain that requires multiples forms, the relations between forms may be required. For instance, different concept values in the first form may activate different types of subsequent forms. For the type of dialog that the necessary domain information is not directly reflected in the conversation, a more complex dialog structure which also models unobservable aspects of the dialog may be required. A teaching strategy in a tutoring dialog is one example of the information that cannot be observed directly from the conversation. Different teaching strategies could be applied to the same question depended on a student's ability. Even though it is possible to extend the form-based dialog structure representation to account for more types of dialogs, other existing dialog representations may be more suitable than forms for some particular domains and applications. Learnability of an additional component is also subjected to future research.

7.3.2 Improving the performance of dialog structure acquisition algorithms

The performance of dialog structure acquisition algorithms can be improved along several dimensions.

7.3.2.1 Improving the proposed unsupervised learning algorithms

All of the unsupervised learning approaches presented in this thesis still have room for improvement. Typically, adding more useful features and combining different learning algorithms together can improve the learning performance.

For a concept identification problem, more efficient concept word selection criteria could be identified by adding new concept word indicators and by combining different types of criteria together. One additional type of indicator that might be worth experimenting is a name entity flag (whether a word is a name entity or not). A word that

[1] Executing a domain action can be considered as a specific case of a segment purpose.

is classified as a name entity (for example, location or time expression) is likely to capture domain information.

For a dialog segmentation problem, both TextTiling and HMM-based segmentation algorithms, which rely on lexical similarity between dialog segments, have some difficulty identifying the boundary between two sub-tasks of the same type. Lexical similarity, which is an efficient feature for identifying the boundary between two sub-tasks that belong to dissimilar form types, may not provide enough information for determining the boundary between two instances of the same form type since their contents are more similar. To solve this problem, additional features that are not sensitive to the content of the segment, such as a prosodic feature, are necessary. It has been shown that prosodic features are able to identify a boundary between the segments that contain quite similar contents (Swerts and Ostendorf, 1997). Since inaccurate segment boundaries affected the performance of the clustering algorithm, a better dialog segmentation algorithm will not only improve the quality of the segments but also the quality of a sub-task clustering results.

To reduce the number of splitting in frequent sub-tasks in a sub-task clustering problem, different criteria for choosing a cluster to split at each iteration can be experimented. For example, a criterion based on the overall similarity of a cluster could be used instead of the one that always chooses to split the largest cluster. A more efficient clustering algorithm can also improve the performance of the HMM-based segmentation algorithm since it provides a better state representation that can differentiate between dialog segments that belong to dissimilar sub-tasks.

The advantages of different algorithms can be combined to improve the performance of the learning algorithms. For both concept identification and sub-task boundary detection problems, competitive approaches seem to have complementary advantages. For concept identification, the statistical approach and the knowledge-based approach have different strong points. The statistical approaches while able to capture domain-specific usage of concept words cannot accurately identify infrequent concept words due to a sparse data problem. The knowledge-based approach, on the other hand, can identify domain concepts very accurately, but on the condition that the concepts are present in the knowledge source. One possible combination method is to acquire an initial set of concepts through a statistical clustering approach then revise these initial concepts with a knowledge-based clustering approach. A statistical clustering approach allows us to recover as many potential concepts as possible while a knowledge-based clustering approach can improve the quality of the initial concepts by adding missing concept members and removing incorrect concept members. For sub-task boundary detection, the TextTiling and the HMM-based segmentation algorithm are good at different types of boundaries. The HMM-based algorithm performs better on very fine-grained segments boundaries while the TextTiling algorithm performs better on the boundaries between consecutive sub-tasks of the same type. Therefore, combining the predictions made by both algorithms could help improve sub-task boundary detection performance.

Since tasks, sub-tasks, and concepts are the components of the same dialog structure, information about one component may be useful for inferring another component. As shown in Chapter 6, information from concept annotation can improve both dialog segmentation and sub-task clustering results. Information from sub-task boundaries and types of tasks and sub-tasks should also help improve the performance of a concept identification algorithm. For example, based on an assumption that a list of concepts occurs in one sub-task is distinguishable from a list of concepts occurs in other sub-tasks, a similarity score between words that occur in the same type of sub-task should receive more weight than a similarity score between words that occur in different types of sub-tasks. More interaction between components such as an iterative approach should also be considered. To measure the effect of propagated errors when incorporating information from another dialog structure component in the learning algorithm, the performance when incorporating predicted components and the performance when incorporating hand-annotated components should be compared.

7.3.2.2 Learning from a larger corpus

The unsupervised learning approaches investigated in this thesis have been applied to dialog corpora that are quite small. The limitation came from the amount of annotated data. Even though all the learning algorithms are unsupervised, reference annotation is still required in order to evaluate the performance of the proposed algorithms. More dialog data should improve learning accuracy as it reduces a sparse data problem that the statistical learning approaches used in this thesis have encountered. Unsupervised learning algorithms proposed in this thesis and an efficient annotation tool can expedite the annotation process. Human-annotated data can be obtained by correcting the form-based dialog structure components that the learning algorithm predicted. It is also interesting to see the performance gain each learning algorithm can achieve when more dialog data is available. In addition to the learning accuracy on a specific data set, different learning algorithms can be compared in terms of the steepness in their learning curve.

7.3.2.3 Exploring the use of supervised learning algorithms

The investigation in this thesis focuses mainly on unsupervised learning algorithms as the target domain-specific information (i.e. a list of concepts and sub-tasks) has not been specified and needs to be inferred from in-domain dialogs. Nevertheless, for a comparison purpose, it is interesting to see how well an unsupervised learning approach performs compared to a supervised learning approach on the same dialog structure acquisition task.

Some supervised learning techniques can be applied to the domain knowledge acquisition problem if these learning algorithms are trained on dialog data from a different domain, or if the target dialog structure components are not utilized in the training. For example, a supervised segmentation algorithm trained on an air travel domain can be used to identify sub-task boundaries of dialogs in a meeting domain. Nevertheless, the performance of this supervised algorithm is also depended on the similarity between the domain that it is trained from and the domain that it is applied to. If the characteristics of both domains are similar and the learning algorithm uses the features which are a good indicator in both domains, the performance of the algorithm on both domains should not be much different. On the other hand, if the characteristics of both domains are different, cross-domain performance may not be as good due to the mismatch. It is more difficult to apply a supervised learning algorithm to the clustering task across domains than to apply it to the segmentation task. In addition to the possible different in domain characteristics, the sets of labels (e.g. the list of concepts) are also different. Moreover, some concepts that look similar might have different roles in different domains. For instance, the numbers “one” to “nine” are **quantity** in one domain but are **hour_number** in another domain.

For the problem of concept clustering, it can also be considered as coordinate term classification (coordinate terms are words that have the same hypernym) and can utilize some existing supervised algorithms. A coordinate term classifier (Snow et al., 2005, 2006) was trained on information in the WordNet lexicon database to identified coordinate terms in a given data set such as newswire. By using this algorithm we should be able to identify coordinate terms, which are words that belong to the same concept, from both in-domain conversations and additional text corpora which are related to the domain of interest such as related information from the web. Other supervised learning approaches such as named entity extraction and semantic role labeling which were trained on standard corpora, for example, PropBank (Palmer et al., 2005) for semantic role labeling, can be used to extract useful features for concept identification and clustering. For instance, words that are classified as the same entity type, or words that have the same semantic role for the same verb are more likely to belong to the same concept.

A semi-supervised learning approach trained on in-domain data might be more efficient than a supervised learning approach trained on cross-domain data. For a semi-supervised learning, a small number of annotated instances are provided. For instance, a couple of concept instances can be annotated and then use as seeds in the concept clustering algorithm. The algorithm proposed by Zhu et al. (2003) is one example of a semi-supervised algorithm. For information-accessing tasks, information from the backend database is a very useful resource for acquiring domain-specific information. However, the design of the database, such as the list of fields, may not correspond perfectly to the domain-specific information actually communicated in human-human conversations. The information from the database can be used as annotated instances for a semi-supervised learning approach.

Incorporating human feedback into the learning process should also improve the learning performance. Unsupervised learning can be used first to explore unannotated data. If additional knowledge from a human is required, active learning will be applied to select the most informative questions. This approach will require only a minimal amount of human annotation.

7.3.3 Implementing a dialog system from acquired domain knowledge

Since the data-driven approach presented in this thesis could potentially reduce human effort in developing a new task-oriented dialog system, it should be implemented in a practical dialog system. In order to do so, the learning approaches of all dialog structure components need to be put together into a completed learning system. A human should also be included in the loop to correct any learning mistake that the system might produce. This process includes putting all the learning approaches into a single framework; streamlining all the parts that require human supervision and providing a helpful interface for annotating the data; presenting the result of the learning system in a systematic way and optionally allowing a human to revise the result. This may require a slight modification on the learning approaches to make all of them fit together.

References

K. Aas and L. Eikvil. 1999. *Text Categorisation: A Survey,* Technical Report NR 941, Norwegian Computing Center.

S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In E. Black (Ed.), *Proceedings of a workshop on Speech and natural language,* pp. 306-311. Pacific Grove, California, United States: Morgan Kaufmann Publishers Inc.

J. Alexandersson. 1995. Plan Recognition in Verbmobil. In *Proceedings of the IJCAI-95 Workshop on the Next Generation of Plan Recognition Systems*. Montreal, Canada.

J. Alexandersson, E. Maier, and N. Reithinger. 1995. A robust and efficient three-layered dialogue component for a speech-to-speech translation system. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics,* pp. 188-193. Dublin, Ireland: Morgan Kaufmann Publishers Inc.

J. Alexandersson and N. Reithinger. 1995. Designing the Dialogue Component in a Speech Translation System A Corpus Based Approach. In *Proceedings of the 9th Twente Workshop on Language Technology: Corpus-Based Approaches to Dialogue Modelling (TWLT9),* pp. 35-43. Enschede, the Netherlands.

J. Alexandersson and N. Reithinger. 1997. Learning Dialogue Structures From A Corpus. In *Proceedings of EuroSpeech-97*. Rhodes, Greece.

J. F. Allen and C. R. Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence,* 15:143-178.

J. F. Allen, L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. G. Martin, B. W. Miller, M. Poesio, and a. D. R. Traum. 1995. The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence,* 7(1):7 - 48.

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

J. Arguello and C. P. Rosé. 2006. Topic Segmentation of Dialogue. In *Proceedings of Workshop on Analyzing Conversations in Text and Speech*.

N. Asher. 1993. *Reference to Abstract Objects in Discourse*. Dordrecht, the Netherlands: Kluwer Academic Publishers.

H. Aust, M. Oerder, F. Seide, and V. Steinbiss. 1995. The Philips automatic train timetable information system. *Speech Communication,* 17(3-4):249-262.

S. Banerjee, J. Cohen, T. Quisel, A. Chan, Y. Patodia, Z. A. Bawab, R. Zhang, A. Black, R. Stern, R. Rosenfeld, A. I. Rudnicky, P. Rybski, and M. Veloso. 2004. Creating Multi-Modal, User-Centric Records of Meetings with the Carnegie Mellon Meeting Recorder Architecture. In *Proceedings of the ICASSP 2004 Meeting Recognition Workshop*. Montreal, Canada.

S. Banerjee and A. I. Rudnicky. 2006. You Are What You Say: Using Meeting Participants' Speech to Detect their Roles and Expertise. In *the NAACL-HLT 2006 workshop on Analyzing Conversations in Text and Speech*. New York, NY.

S. Bangalore, G. D. Fabbrizio, and A. Stent. 2006. Learning the Structure of Task-Driven Human-Human Dialogs. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

S. Bangalore, D. Hakkani-Tür, and G. Tur. 2006. Introduction to the Special Issue on Spoken Language Understanding in Conversational Systems. *Speech Communication: Special Issue on Spoken Language Understanding,* 48(3-4):233-238.

R. Barzilay and L. Lee. 2004. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference,* pp. 113-120. Boston, MA.

D. Beeferman, A. Berger, and J. Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning,* 34(1-3):177-210.

C. Bennett, A. F. Llitjós, S. Shriver, A. Rudnicky, and A. W. Black. 2002. Building Voicexml-based applications. In *Proceedings of ICSLP-2002*. Denver, Colorado.

E. Bilange. 1991. A task independent oral dialogue model. In *Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics,* pp. 83-88. Berlin, Germany: Association for Computational Linguistics.

D. Bohus and A. Rudnicky. 2002. LARRI: A Language-Based Maintenance and Repair Assistant. In *Proceedings of IDS-2002*. Kloster Irsee, Germany.

D. Bohus and A. I. Rudnicky. 2003. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Proceedings of Eurospeech2003*. Geneva, Switzerland.

T. Brants. 2000. Inter-Annotator Agreement for a German Newspaper Corpus. In *Proceedings of LREC2000*. Athens, Greece.

M. E. Bratman. 1987. *Intention, Plans, and Practical Reason*. Cambirdge, MA: Harvard University Press.

M. E. Bratman, D. J. Israel, and M. E. Pollack. 1988. Plans And Resource-Bounded Practical Reasoning.

E. Brill. 1994. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. Seattle, WA.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.,* 18(4):467-479.

T. Bub and J. Schwinn. 1996. VERBMOBIL: the evolution of a complex large speech-to-speechtranslation system. In *Proceedings of ICSLP-96,* pp. 2371-2374. Philadelphia, PA.

S. Carberry. 1990. *Plan Recognition in Natural Language Dialogue*. Cambridge, MA: MIT Press.

J. Carletta. 1996. Assessing agreement on classification tasks the kappa statistic. *Computational Linguistics*, 22(2):249-254.

J. Carletta, A. Isard, S. Isard, J. C. Kowtko, G. Doherty-Sneddon, and A. H. Anderson. 1996. *HCRC Dialogue Structure Coding Manual*.

J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13-31.

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics,* pp. 310-318. Santa Cruz, California: Association for Computational Linguistics.

A. Chotimongkol and A. Rudnicky. 2002. Automatic Concept Identification in Goal-Oriented Conversations. In *Proceedings of ICSLP 2002*. Denver, Colorado.

G. E. Churcher, E. S. Atwell, and C. Souter. 1997. *Dialogue Management Systems: a Survey and Overview*, Technical Report 97.6, School of Computer Studies, University of Leeds.

M. Civit, A. Ageno, B. Navarro, N. Bufi, and M. A. Marti. 2003. Qualitative and quantitative analysis of annotators' agreement in the development of Cast3LB. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*. Vaxjo, Sweden.

P. R. Cohen and C. R. Perrault. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science,* 3:177-212.

R. Cole, J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, A. Zampolli, and V. Zue. 1997. *Survey of the state of the art in human language technology*: Cambridge University Press.

P. C. Constantinides, S. Hansma, C. Tchou, and A. I. Rudnicky. 1998. A schema-based approach to dialog control. In *Proceedings of ICSLP-1998.*

M. Core and J. Allen. 1997. Coding Dialogs with the DAMSL Annotation Scheme. In *AAAI Fall Symposium on Communicative Action in Humans and Machines*. Boston, MA.

D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. 1992. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 318-329. Copenhagen, Denmark: ACM Press.

I. Dagan, L. Lee, and F. Pereira. 1999. Similarity-based models of word cooccurence probabilities. *Machine Learning,* 34(1):43-69.

N. Dahlbäck, A. Jönsson, and L. Ahrenberg. 1993. Wizard of Oz Studies -- Why and How. *Knowledge-Based Systems,* 6(4):258-266.

M. H. Degroot. 1986. *Probability and Statistics* (Second ed.): Addison Wesley.

M. Denecke and A. Waibel. 1997. Dialogue Strategies Guiding Users to their Communicative Goals. In *Proceedings of Eurospeech97,* pp. 1339-1342. Rhodes,Greece.

R. Dhillon, S. Bhagat, H. Carvey, and E. Shriberg. 2004. *Meeting Recorder Project: Dialog Act Labeling Guide,* Technical Report TR-04-002, International Computer Science Institute.

H. Dybkjær, L. Dybkjær, and N. O. Bernsen. 1995. Design, Formalisation and Evaluation of Spoken Language Dialogue. In *Proceedings of the 9th Twente Workshop on*

Language Technology: Corpus-Based Approaches to Dialogue Modelling (TWLT9), pp. 67-82. Enschede, the Netherlands.

M. Eskenazi, A. Rudnicky, K. Gregory, P. Constantinides, R. Brennan, C. Bennett, and J. Allen. 1999. Data Collection and Processing in the Carnegie Mellon Communicator. In *Proceedings of Eurospeech 1999*. Budapest, Hungary.

B. D. Eugenio, P. W. Jordan, and L. Pylkkänen. 1998. *The COCONUT project: Dialogue Annotation Manual*, ISP Technical Report 98-1. Pittsburgh, Pennsylvania, University of Pittsburgh.

J. Feng, S. Bangalore, and M. Rahim. 2003. WebTalk: mining Websites for automatically building dialog systems. In *IEEE ASRU '03*, pp. 168-173. St. Thomas, U.S. Virgin Islands.

G. Ferguson, J. Allen, and B. Miller. 1996. Trains-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*. Edinburgh, Scotland.

A. Ferrieux and M. D. Sadek. 1994. An Efficient Data-Driven Model for Cooperative Spoken Dialogue. In *Proceedings of ICSLP 1994*. Yokohama, Japan.

M. Finke, M. Lapata, A. Lavie, L. Levin, L. M. Tomokiyo, T. Polzin, K. Ries, A. Waibel, and K. Zechner. 1998. CLARITY: Inferring Discourse Structure from Speech. In *Proceedings of Workshop on Applying Machine Learning to Discourse Processing. AAAI-1998 Spring Symposium Series*.

D. H. Fisher. 1987. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2(2):139-172.

G. Flammia and V. Zue. 1995. Empirical evaluation of human performance and agreement in parsing discourse constituents in spoken dialogue. In *Proceedings of Eurospeech 1995*. Madrid, Spain.

R. Freedman. 2000. Plan-based dialogue management in a physics tutor. In *Proceedings of the sixth conference on Applied natural language processing*, pp. 52-59. Seattle, Washington.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Vol. 1, pp. 562-569. Sapporo, Japan: Association for Computational Linguistics.

A. Garland, N. Lesh, and C. Sidner. 2001. Learning Task Models for Collaborative Discourse. In *Proceedings of Workshop on Adaptation in Dialogue Systems, NAACL '01*. Pittsburgh, Pennsylvania.

M. P. Georgeff and F. F. Ingrand. 1989. Decision-Making in an Embedded Reasoning System. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Detroit, MI.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: telephone speech corpus for research and development. In *Proceedings of ICASSP'92*, pp. 517-520.

J. C. Gorman, N. J. Cooke, P. W. Foltz, P. A. Kiekel, and M. J. Martin. 2003. Evaluation of Latent Semantic Analysis-based measures of team communications content. In

Proceedings of the Human Factors and Ergonomic Society 47th Annual Meeting, (HFES 2003).

D. Gross, J. F. Allen, and D. R. Traum. 1993. *The TRAINS 91 Dialogues,* TRAINS Technical Note 92-1, Computer Science Department, University of Rochester.

B. J. Grosz. 1978. Discourse analysis. In D. Walker (Ed.), *Understanding Spoken Language,* pp. 235-268. North-Holland, New York: Elsevier.

B. J. Grosz and S. Kraus. 1996. Collaborative plans for complex group action. *Artificial Intelligence,* 86(2):269-357.

B. J. Grosz, M. E. Pollack, and C. L. Sidner. 1989. Discourse. In Michael I. Posner (Ed.), *Foundations of cognitive science,* pp. 437-467. The MIT Press.

B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics,* 12(3):175-204.

B. J. Grosz and C. L. Sidner. 1990. Plans for Discourse. In Philip R. Cohen and Jerry L. Morgan (Eds.), *Intentions in Communication,* pp. 417-444. Cambridge, MA: MIT Press.

N. Gupta, G. Tür, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Rahim. 2006. The AT&T Spoken Language Understanding System. *IEEE Transactions on Speech and Audio Processing,* 14(1):213-222.

H. Hardy, K. Baker, H. Bonneau-Maynard, L. Devillers, S. Rosset, and T. Strzalkowski. 2003. Semantic and Dialogic Annotation for Automated Multilingual Customer Service. In *Proceedings of Eurospeech 2003*. Geneva, Switzerland.

H. Hardy, A. Biermann, R. B. Inouye, A. Mckenzie, T. Strzalkowski, C. Ursu, N. Webb, and M. Wu. 2004. Data-Driven Strategies for an Automated Dialogue System. In *Proceedings of ACL '04*. Barcelona, Spain.

A. G. Hauptmann and A. I. Rudnicky. 1988. Talking to computers: an empirical investigation. *International Journal of Man-Machine Studies,* 28(6):583-604.

M. A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics,* 23(1):33-64.

J. Hirschberg and D. Litman. 1993. Empirical studies on the disambiguation of cue phrases. *Computational Linguistics,* 19(3):501-530.

J. R. Hobbs. 1996. On the Relation between the Informational and Intentional Perspectives on Discourse. In Eduard Hovy and Donia Scott (Eds.), *Computational and Conversational Discourse: Papers from the NATO Advanced Research Working Group on Burning Issues in Discourse,* pp. 139–157. Berlin: Springer Verlag.

A. Jönsson and N. Dahlbäck. 1988. Talking to a Computer is not Like Talking to Your Best Friend. In *Proceedings of The first Scandinavian Conference on Artificial Intelligence*. Tromsø, Norway.

D. Jurafsky, R. Bates, N. Coccaro, R. Martin, M. Meteer, K. Ries, E. Shriberg, A. Stolcke, P. Taylor, and C. V. Ess-Dykema. 1997. Automatic Detection of Discourse Structure for Speech Recognition and Understanding. In *Proceedings of IEEE Workshop on Speech Recognition and Understanding*. Santa Barbara.

H. Kamp. 1981. A Theory of Truth and Semantic Representation. In J. Groenendijk, T. Janssen and M. Stokhof (Eds.), *Formal Methods in the Study of Language,* pp. 277-322. Amsterdam: Mathematisch Centrum.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer.

M. Karahan, D. Hakkani-Tür, G. Riccardi, and G. Tur. 2003. Combining Classifiers for Spoken Language Understanding. In *Proceedings of ASRU-2003, 8th biannual IEEE workshop on Automatic Speech Recognition and Understanding*. U.S. Virgin Islands.

P. Kingsbury, S. Strassel, C. McLemore, and R. McIntyre. 1997. CALLHOME American English Transcripts. Philadelphia: Linguistic Data Consortium.

I. Kruijff-Korbayová, E. Karagjiosova, K. J. Rodríguez, and S. Ericsson. 2003. A Dialogue System with Contextually Appropriate Spoken Output Intonation. In *Proceedings of the Demo Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*. Budapest, Hungary.

H.-K. J. Kuo and C.-H. Lee. 2001. Simplifying Design Specification For Automatic Training Of Robust Natural Language Call Router. In *Proceedings of ICASSP-2001*. Salt Lake City,. Utah.

L. Lambert and S. Carberry. 1991. A tripartite plan-based model of dialogue. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics,* pp. 47-54. Berkeley, California: Association for Computational Linguistics.

L. Lambert and S. Carberry. 1992. Modeling negotiation subdialogues. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics,* pp. 193-200. Newark, Delaware: Association for Computational Linguistics.

B. Larsen and C. Aone. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 16-22. San Diego, California, United States: ACM Press.

L. B. Larsen and A. Baekgaard. 1994. Rapid Prototyping of a Dialogue System Using a Generic Dialogue Development Platform. In *proceedings of ICSLP-1994,* pp. 919-922. Yokohama, Japan.

S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering,* 6(3-4):323-340.

L. Levin, A. Thyme-Gobbel, A. Lavie, K. Ries, and K. Zechner. 1998. A Discourse Coding Scheme for Conversational Spanish. In *Proceedings of ICSLP-98,* pp. 2335-2338. Sydney, Australia.

G.-A. Levow. 2004. Prosodic Cues to Discourse Segment Boundaries in Human-Computer Dialogue. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue,* pp. 93-96. Boston, MA.

I. Lewin, M. Russell, D. Carter, S. Browning, K. Ponting, and S. Pulman. 1993. A speech-based route enquiry system built from general-purpose components. In

Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech 1993). Berlin, Germany.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics (COLING-ACL),* Vol. 2, pp. 768-774. Montreal, Quebec, Canada: Association for Computational Linguistics.

D. Litman and J. Allen. 1987. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science,* 11(2):163-200.

K. E. Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525-572.

W. C. Mann and S. A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text,* 8(3):243-281.

D. Marcu. 1999. A decision-based approach to rhetorical parsing. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics,* pp. 365-372. College Park, Maryland: Association for Computational Linguistics.

D. Marcu, E. Amorrortu, and M. Romera. 1999. Experiments in constructing a corpus of discourse trees. In *Proceedings of ACL Workshop on Standards and Tools for Discourse Tagging,* pp. 48-57. College Park, MD.

M. F. McTear. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys,* 34(1):90-169.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography,* 3(4):235-244.

J.-U. Möller. 1998. Using unsupervised learning for engineering of spoken dialogues. In *Proceedings of AAAI 1998 Spring Symposium on Applying Machine Learning to Discourse Processing*.

J. D. Moore and M. E. Pollack. 1992. A problem for RST: the need for multi-level discourse analysis. *Comput. Linguist.,* 18(4):537-544.

J. D. Moore and P. Wiemer-Hastings. 2003. Discourse in Computational Linguistics and Articial Intelligence. In Arthur C. Graesser, Morton Ann Gernsbacher and Susan R. Goldman (Eds.), *Handbook of Discourse Processes,* pp. 439-487. Lawrence Erlbaum Associates.

M. Moser and J. D. Moore. 1997. On the correlation of cues with discourse structure: Results from a corpus study. In. Submitted for publication.

National Institute of Standards and Technology (NIST). 1998. The Topic Detection and Tracking Phase 2 (TDT2) Evaluation Plan Version 3.7.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Comput. Linguist.,* 31(1):71-106.

A. Pargellis, E. Fosler-Lussier, A. Potamianos, and C.-H. Lee. 2001. Metrics for Measuring Domain Independence of Semantic Classes. In *The proceedings of the Seventh European Conference on Speech Communication and Technology (Eurospeech '01)*. Aalborg, Denmark.

L. Polanyi. 1996. *The Linguistic Structure of Discourse*, Technical Report CSLI-96-200. Stanford CA, Center for the Study of Language and Information, Stanford University.

M. E. Pollack. 1992. The uses of plans. *Artificial Intelligence,* 57(1):43-68.

M. Purver, P. Ehlen, and J. Niekrasz. 2006. Detecting action items in multi-party meetings: Annotation and initial experiments. In Steve Renals, Samy Bengio and Jonathan Fiscus (Eds.), *Machine Learning for Multimodal Interaction: Third International Workshop, MLMI 2006, Bethesda, MD, USA, May 1-4, 2006, Revised Selected Papers,* Vol. 4299, pp. 200-211. Bethesda, MD: Springer.

E. Rasmussen. 1992. Clustering Algorithms. In William B. Frakes and Ricardo Baeza-Yates (Eds.), *Information Retrieval: Data Structures & Algorithms*. Prentice Hall.

A. Raux, B. Langner, A. W. Black, and M. Eskenazi. 2003. LET'S GO: Improving Spoken Dialogue Systems for the Elderly and Non-natives. In *Proceedings of Eurospeech-2003*. Geneva, Switzerland.

G. Riccardi and S. Bangalore. 1998. Automatic Acquisition of Phrase Grammars for Stochastic Language Modeling. In *Proceedings Of the 6th Workshop on Very Large Corpora,* pp. 186-198. Montreal, Canada.

C. Rich, C. L. Sidner, and N. Lesh. 2001. Collagen: applying collaborative discourse theory to human-computer interaction. *AI Magazine,* 22(4):15-25.

C. P. Rosé, D. Bhembe, S. Siler, R. Srivastava, and K. VanLehn. 2003. The Role of Why Questions in Effective Human Tutoring. In *Proceedings of Artificial Intelligence in Education 2003*. Sidney , Australia .

C. P. Rosé, B. D. Eugenio, L. S. Levin, and C. V. Ess-Dykema. 1995. Discourse processing of dialogues with multiple threads. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics,* pp. 31-38. Cambridge, Massachusetts: Association for Computational Linguistics.

C. P. Rosé and B. S. Hall. 2004. A Little Goes a Long Way: Quick Authoring of Semantic Knowledge Sources for Interpretation. In *Proceedings of the Second International Workshop on Scalable Natural Language Understanding*. Boston, MA.

A. I. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, X. W., and A. Oh. 1999. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Proceedings of Eurospeech 1999*. Budapest, Hungary.

M. D. Sadek, P. Bretier, and F. Panaget. 1997. ARTIMIS: Natural Dialogue Meets Rational Agency. In *Proceedings of 15th International Joint Conference on Artificial Intelligence, (IJCAI-97),* pp. 1030–1035. San Francisco, CA: Morgan Kaufmann Publishers.

J. R. Searle. 1975. A taxonomy of illocutionary acts. In Keith Gunderson (Ed.), *Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Science,* Vol. VII. Minneapolis: University of Minnesota Press.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys,* 34(1):1-47.

J. M. Sinclair and M. Coulthard. 1975. *Towards an analysis of Discourse: The English used by teachers and pupils*: Oxford University Press.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing Dialogue Managment with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research,* 16:105-133.

K.-C. Siu and H. M. Meng. 1999. Semi-Automatic Acquisition of Domain-Specific Semantic Structures. In *Proceedings of Eurospeech-1999*. Budapest, Hungary.

R. W. Smith and D. R. Hipp. 1994. *Spoken natural language dialog systems: a practical approach*. New York, NY: Oxford University Press.

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of Advances in Neural Information Processing Systems*. Vancouver, Canada.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL,* pp. 801-808. Sydney, Australia: Association for Computational Linguistics.

B. Snyder and M. Palmer. 2004. The English all-words task. In *Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pp. 41-43. Barcelona, Spain.

M. Steinbach, G. Karypis, and V. Kumar. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

A. Stent. 2000. Rhetorical Structure in Dialog. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'2000).*

A. Stent, R. Prasad, and M. Walker. 2004. Trainable Sentence Planning for Complex Information Presentation in Spoken Dialog Systems. In *Proceedings of ACL 2004*. Barcelona, Spain.

A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. V. Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin, and M. Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics,* 26(3):339-373.

M. Swerts and M. Ostendorf. 1997. Prosodic and lexical indications of discourse structure in human-machine interactions. *Speech Communication,* 22(1):25-41.

M. Taboada and W. C. Mann. 2006. Rhetorical Structure Theory: looking back and moving ahead. *Discourse Studies,* 8(3):423-459.

D. R. Traum, P. Bohlin, J. Bos, S. Ericsson, S. Larsson, I. Lewin, C. Matheson, and D. Milward. 2000. *Dialogue Dynamics and Levels of Interaction*, Technical Report Trindi Project Deliverable D3.1, TRINDI.

D. R. Traum and E. A. Hinkelman. 1992. Conversation Acts in Task-Oriented Spoken Dialogue. *Computational Intelligence,* 8(3):575--599.

D. Tsovaltzi and E. Karagjosová. 2004. A View on Dialogue Move Taxonomies for Tutorial Dialogues. In *Proceedings of The 5th SIGdial Workshop on Discourse and Dialogue*. Cambridge, MA.

G. Tür, A. Stolcke, D. Hakkani-Tür, and E. Shriberg. 2001. Integrating prosodic and lexical cues for automatic topic segmentation. *Computational Linguistics,* 27(1):31-57.

D. Vilar, M. J. Castro, and E. Sanchis. 2003. Connectionist Classification and Specific Stochastic Models in the Understanding Process of a Dialogue System. In *Proceedings of Eurospeech 2003*.

M. Woszczyna and A. Waibel. 1994. Inferring linguistic structure in spoken language. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

J. P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. v. Mulbregt. 1998. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing,* Vol. 1, pp. 333-336. Seattle, WA.

Y. Yang and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 42-49. Berkeley, California, United States: ACM Press.

N. Yankelovich. 1997. Using Natural Dialogs as the Basis for Speech Interface Design. In Susann Luperfoy (Ed.), *Automated Spoken Dialog Systems*. Cambridge, MA: MIT Press.

X. Zhu, J. Lafferty, and Z. Ghahramani. 2003. Combining Active Learning and Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.

V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. 2000. JUPITER: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing,* 8(1):85-96.