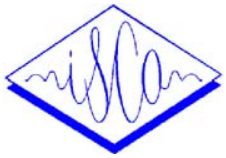
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

A Multi-Layer Conversation Management Approach for Information Seeking Applications

Shimei Pan

IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY. USA 10532
Shimei@us.ibm.com

10.21437/Interspeech.2004-129

Abstract

The paper describes a new conversation management approach which may apply to a class of mixed initiative information seeking applications. It employs an application-independent conversation model inspired by the conversation theory of Grosz and Sidner [1]. Based on the model, we design a multi-layer conversation manager which employs instance-based learning (IBL) to determine conversation plans, while performs modularized realization of each conversation move in a conversation plan. Because we adopt IBL to determine conversation plans, new conversation behaviors are easy to incorporate. Our realization of conversation moves is also application-independent. We illustrate this method with a real estate application.

1. Introduction

In an information seeking application, a conversation manager needs to perform many tasks such as assisting data retrieval and data presentation, assisting data exploration and navigation and handling exceptions. To accomplish these tasks, some conversation systems adopt finite-state machine (FSM)-based approaches [2] in which control decisions are encoded in the FSM itself. Since these approaches have exponential complexity, it's hard for them to handle complex information seeking tasks. A similar but more flexible approach makes conversation decisions based on form filing [3]. The slots in a form specify information to be obtained to complete a task such as data query. Since dialogue complexity is limited by form design, sophisticated control strategies are hard to accommodate. Unlike the FSM and Form-based approaches, a plan-based conversation manager treats conversations as collaborative planning processes [4,5,6]. Because plan-based approaches are founded on a model of planning and fulfilling communicative goals, they are more flexible than the FSM and form-based approaches. However, because it employs expensive AI planning and plan recognition techniques, the computation is combinatorially intractable [7,8,9].

In this paper, we propose a flexible and yet efficient conversation management approach that employs instance-based learning (IBL) to decide conversation plans while performs modularized realization of conversation moves. It employs a representation of *conversation history*, which records all the past and present conversation moves by all the participants. Since other than the conversation history, there is no task representation in the conversation manager, this approach is flexible and domain-independent. Unlike the plan-based approaches which employ computationally expensive AI planning techniques, we do not explicitly use planning. Instead, we use IBL as the main decision making mechanism. Since IBL is a pattern based machine learning method that makes current decisions based on previous experience, it is more efficient than the plan-based approaches. We also separate high-level decisions on deliberating conversation plans from detailed decisions on modelling conversation moves. This multi-layer design makes it easier to incorporate new conversation plans. Moreover, we model each conversation move in a separate function module. Because new function modules can be plugged in easily, this modularized design improves a system's flexibility and extensibility.

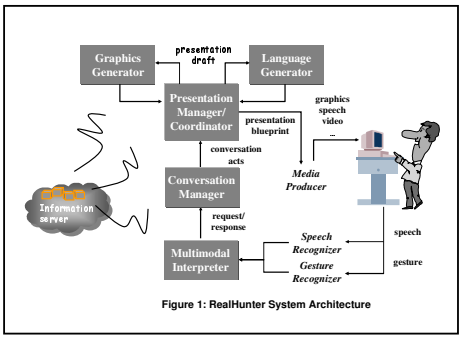


Figure 1: RealHunter System Architecture

In the rest of the paper, we first introduce a testbed system called Real Hunter. Then, in section 3, we focus on the representation of the conversation history because it is the main information carrier for reasoning and passing conversation decisions. In section 4, we illustrate a multi-layer conversation manager. Finally, we discuss results and draw conclusions.

2. The Application

Our approach is embodied in a testbed system called Real Hunter, which helps users search residential real estate information through multimodal conversations. Figure 1 shows Real Hunter's main components. A user can interact with Real Hunter using multiple input channels, such as speech and gesture. Then, the multimodal interpreter exploits various contexts to produce an interpretation that captures the meanings of user inputs. Based on the interpretation, the conversation manager decides how the system should act/react by generating a set of conversation moves. At the same time, it also updates the conversation history so that decisions can be passed to the next component. Upon receiving the conversation history, the presentation manager sketches a presentation draft that expresses the outline of a multimedia presentation. Based on this draft, the language and the graphics generator work together to author a multimedia blueprint which is then sent to a media producer to be realized. To support all the components described above, the information server supplies various contextual information, including domain data, a conversation history, a user model, and a presentation environment model. In the following, we focus on the

conversation manager. We start with the representation of conversation history.

3. Conversation History

Our conversation history representation is inspired by the theory of [1]. There are three types of node in the conversation history: *conversation segment*, *conversation unit* and *conversation act*. The conversation segment aggregates system and user turns into a hierarchical structure. Each segment has a segment purpose called *segment intention*. Each segment may have embedded *sub-segments*. Each sub-segment has its own segment intention. In addition, each segment has one *user unit* and one *system unit*, corresponding to a user or system turn. Each user/ system unit has one or more *user/system acts*. Each user/system act has its own purpose called *act intention*. Each act also has an *attention* indicating participants' current focus space. *Attentions* can be either a *simple attention* or a *complex attention.* A simple attention describes simple data items while a complex attention describes a set, list, relation, predicate, action, etc. In addition to conversation nodes, the conversation history also includes a *presentation draft* and a *presentation blueprint.* Both of them provide essential information for discourse interpretation, especially, reference resolution.

4. Multi-layer Conversation Management

The multi-layer conversation manager has two components: an IBL-based *conversation strategy manager* and a *conversation move realizer* (shown in Figure 2). The IBL strategy manager deliberates a conversation plan that contains a sequence of conversation moves. Each conversation move later will be realized by the conversation move realizer as either a conversation segment or act to be added to the conversation history or as an action to be performed by a *transaction manager* to update application data. Most conversation context information used for IBL learning and conversation move realization is from the conversation history. Other contextual information, including characteristics of the retrieved data (domain data), knowledge on the operation and presentation environment (presentation preferences), knowledge about the current user (user preferences), as well as parameters maintained inside the conversation manager (conversation status), play important roles in supporting decision making. In the following, we present the details of the IBL strategy manager and the conversation move realizer.

4.1. IBL Strategy Manager

To illustrate the instance-based strategy manager, we start with the instance representation.

4.1.1. Instance Repository

Each instance in the instance repository has a left hand side and a right hand side. The left hand side (the predicting part) is an abstraction of the characteristics of the current conversation context. It is represented as a vector of features. The right hand side (the predicted part) is a conversation plan, represented as a sequence of conversation moves. Both the predicting variables and the predicted conversation plans in an instance are application independent.

Currently, for each predicting vector, we extract 33 features from the context. The majority are extracted from the conversation history, such as the *status* and the *intention* of the last conversation segment, the previous conversation segment (follow a temporal link), and the parent conversation segment (follow a structural link). Similarly, we extract the *status* and *intention* from various system and user acts. In addition to features from the conversation history, we also extract other contextual features that are useful for conversation decisions. For example, features from the user preference model include a user's *experience* (whether he is a new or repeated user) and his *knowledge* about the application domain (whether he is familiar with the area). These user preference features can help the system decide whether a profile needs to be created (for a new user), or whether additional information needs to be presented (if the user does not have certain knowledge). Features from the data model include data volume (how many data entities have been retrieved) and data complexity (the width and depth of each retrieved entity). Both data volume and data complexity are used to determine appropriate presentation forms. For example, if a large amount of data is retrieved; the system may decide to use *summarize* instead of direct *describe*; or in stead of presenting information, it may decide to take the initiative and prompt the user for a specific piece of information. Features from the presentation manager include the system's presentation capability and graphical summarization capability. For example, the system may choose to *summarize* only if either the graphics or the language generator is capable of creating a summary for a data set. The conversation manager also keeps track of a set of conversation parameters such as who has the initiative. One important criterion for selecting predicting variables is that they should be application independent. For example, instead of using domain attributes directly in the predicting vector, we extract domain independent meta information such as data volume and data complexity, so that the instance repository is domain independent.

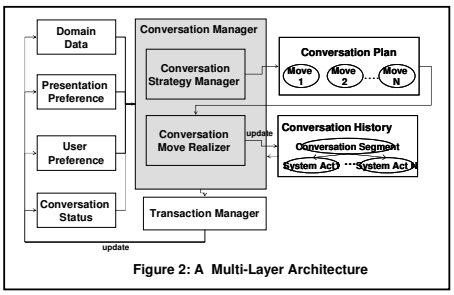


Figure 2: A Multi-Layer Architecture

Typical conversation moves in the right hand side of an instance include *NewSegment(SegIntention), NewAct(ActIntention*) and *Action(Type, Parameter).* Currently, there are 34 segment intentions in Real Hunter, including D*ataAccess_Seek, DataManipulation_Sort, ViewManipulation_GoBack, ExceptionHandling_UnknownInput, DataNavigation_BySpecifiedFeature,* and *Communication_SocialMsg*. There are also 24 act intentions in Real Hunter, including *Present_Describe, Present_Summarize, Request_Inquire, Acknolledge_Appreciation*. Typical action types include *Save, Delete, Create, and Change-Initiative.* Even though the segment and act intentions described above are mainly designed for Real Hunter, all of them are applicable to general information seeking applications. In the following, we describe how instances in the instance repository are used in IBL-based strategy manager.

4.1.2. Instance-based Learning

Each time the conversation manager needs to make a decision, it first extracts the predicting variables from the current conversation context. To decide the best act/response strategy, the conversation manager searches through all the instances in the

instance repository. The one most similar to the current context is selected. To find the most similar instance in the instance repository, the conversation manager dynamically computes the distance between the current context vector and the instances in the instance repository. The final distance is a weighted combination of the distance of each predicting variable. The weight currently is assigned heuristically. For example, since the influence of discourse context decays overtime, the system gives more weight to the features from the current user act, system act, and conversation segment than those from the past history. We also assign higher weight to the segment intention than the act intention. This is because the current segment intention definition is more refined and more informative than that of act intention.

Because we adopt a weighted measure for instance matching, no exact match is required, which decreases the demand for the number of distinct instances in the instance repository. The output of the strategy manager is a sequence of conversation moves. In the following, we describe how conversation moves are realized in Real Hunter.

4.2. Modularized Realization of Conversation Moves

IBL Strategy Manager selects a conversation plan that contains a sequence of conversation moves. A conversation move, however, is only an abstraction of what needs to be done. The details for executing each move are left for the move realizer to decide. For example, if a user does not know what to do, the strategy manager may decide to take the initiative and execute the following actions: "Action(ChangeInitiative, system) NewSegment(NavigateBySpecifiedFeature) and NewAct(Request_Inquire)". Upon receiving these decisions, the move realizer will decide among all the possible features, which feature is the best for navigation. We use a criterion similar to [10] in which conversation efficiency is the main concern. To help a user reach his target the quickest, we select a navigation feature which maximizes the expected search space reduction. The expected search space reduction for a categorical variable is defined as:

$$R_{f_i} = 1 - \sum_x \Pr ob(x).\Pr ob(f_i = x) \quad (1)$$

where *Prob(x)* is the likelihood a user selects *x* when the system prompts for the value of feature f_i and $Prob(f_i=x)$ is the probability distribution of $f_i=x$ in the current search space. Because, $Prob(f_i=x)$ is a measure of the size of the resulting search space, R_{fi} is the average size reduction if f_i is prompted next. Similarly, the expected search space reduction for a continuous variable f_i is:

$$R_{fi-\max} = 1 - \int_{-\infty}^{+\infty} \Pr ob(x).\Pr ob(f_i < x) \quad (2)$$

$$R_{fi-\min} = 1 - \int_{-\infty}^{+\infty} \Pr ob(x).\Pr ob(f_i > x) \quad (3)$$

where *Prob(x)* is the probability a user selects value *x* when the system prompts for the upper and lower end of a range, and $Prob(f_i<x)$, $Prob(f_i>x)$ is the probability that $(f_i<x)$ or $(f_i>x)$ in the current search space. Since the expected search space reduction is only related to the data distribution and a user's preferences (represented as *Prob(x)* in formula (1) (2) and (3)), it is application and task independent.

Once a navigation feature is selected, the system formulates a new conversation segment and act. The corresponding conversation history segment is shown in Figure 3 (in this figure, the navigation feature selected is *AskingPrice*). The *constraint markup* in the act attention indicates that the *constraint* is formulated by the system

Conversation Segment:
Status: *active*
Intention
Type:*Navigation_By_Specified_Feature*
System Unit: Role: *ICP*
Sysem Act
Act Intention:
Satus:*Open* Type: *Request_Inquiry*
Simple Attention
Onto:*Concept*
MetaType:*Constraint*
Constraint:
EntityName: ***AskingPrice***
ConstraintMarkup
source: *Formulated*
role: *To_Navigate*
evaluate: *False*

Figure 3: Conversation History for Navigation

(instead of the user). It is used for *navigation* (not for *evaluation*). Upon receiving this instruction, the presentation manager asks the language or graphics generator to request the value of the selected feature indicated in the *entity name* field in the constraint.

In the above example, in order to formulate a new attention, the *NewAct* module employs an efficiency-based criterion to systematically select a navigation feature. In general, different criteria may be used to select the navigation feature. This modularized design limits the impact of switching to a different navigation approach to a single function module.

4.3. An Example

Figure 4 shows a segment of the conversation history corresponding to "*User*: show me ranches. *System*: Here are all the ranches". In this simple example, after the user submits the query "show me ranches", the interpreter formulates a new conversation segment with a segment intention "*DataAccess_Seek*". It also creates a new user unit. Inside the user unit, it adds one user act with an act intention "*request_inquire*". The attentional space associated with the act is a simple attention. It represents a simple user query for a list of "*SingleResidenceUnits* with their styles equal to *ranch*". The retrieved houses are stored in the *result* field. Given this request, the strategy manager first extracts the predicting feature vector with the current conversation segment intention equals to "*DataAccess_Seek*" and user act intention equals to "*Request_Inqure*". In addition, it also derives the segment, user act and system act intention and status from the past conversation history, data volume and complexity from the data server, user experience and user preferences from the user model, presentation capability from the presentation manager and conversation status from the conversation manager itself. This predicting feature vector is then used to match against the left hand side of all the instances in the instance repository. The one that is most similar to the current vector is selected and the conversation plan in the right hand side of the selected instance is used as the current response strategy. The conversation plan in the selected instance indicates that a system act *NewAct(Present_Describe)* should be formulated. Given this move, one function module, the *NewAct* module, is called to update the conversation history. It first decides how to formulate a system act attention. In this case, the system act attention is a copy of the user act attention because the focus space has not changed. Then the *NewAct* module decides where to

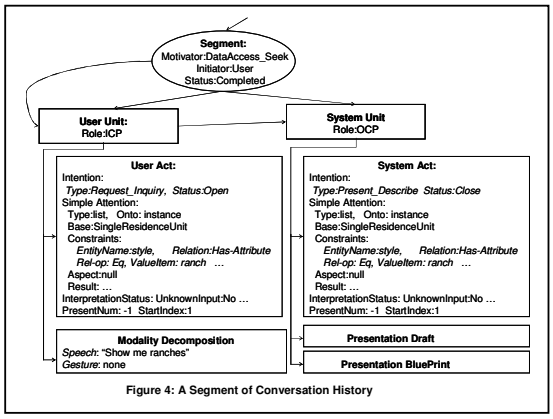


Figure 4: A Segment of Conversation History

add this system act in the conversation history. Since it is a direct response to the request in the last user unit, the new system act is inserted under the same conversation segment as the last user unit. After receiving the updated conversation history, the presentation manager will decide how to formulate a multimedia presentation to describe the set of ranches in the *result* field in the system act attention. It will ask the *language generator* to formulate a sentence "Here are all the ranches". It will also ask the *graphics generator* to show all the retrieved ranches on a map.

5. System Implementation and Results

The proposed approach has been implemented for Real Hunter. To test the usability and robustness of the system, we conducted a user study in which two pilots and six subjects were asked to use the system to complete two information seeking tasks previously tested in our Wizard-of-Oz study. Since the system does not have constraints on how a user should formulate a data query, the resulting conversations demonstrate diverse queries and navigation patterns. In the end, all the participants completed the tasks with an average of five turns. Based on their feedbacks, the users like the system's flexibility in information access. For example, during the user study, the conversation manager can handle complicated user queries such as "show me houses with at least 2 acres of land in a school district with at least 95% high school seniors attending college last year". In addition, the system is also capable of handling various exceptions such as unknown input, and incomplete input, conducting system initiated intelligent data navigation such as query refinement and query relaxation, and generating appropriate social messages such as *solute*, *appreciation, commendation* and *disapproval*.

6. Related Work

The instance-based strategy manager shares some properties with the conversational case-based reasoning (CCBR) framework. CCBR however, was proposed for problem solving tasks [11,12]. Unlike cases in CCBR systems which encode solutions to a task, such as equipment maintenance, instances in our system encode general conversation plans that are task-independent. As a result, our system can support sophisticated conversation behaviors which may apply to different information seeking applications. Various multi-layer conversation management architectures were proposed. [13,14,15]. However, their definitions for layers are quite different from ours. For example, [13] adopted a 2-layer conversation manager. At the bottom layer are conversation games, which typically encode adjacent pairs (such as request, reply). At the top layer, it employs planning to generate a plan with instantiated conversation games as its primitive steps. Unlike [13], we do not explicitly use planning. Instead, we employed IBL as the main decision making mechanism, which is more efficient than AI planning. Moreover, our conversation moves in a conversation plan are task independent, which are different from the instantiated conversation games in [13].

7. Conclusions

In this paper we propose a new conversation management approach which uses IBL to determine general conversation plans while employs modularized realization of conversation moves. Since it is mainly based on a general conversation model, it is more flexible than the FSM and form-based approaches. Since it relies on IBL to determine conversation plans, new conversation behaviours are easy to incorporate. It is also more efficient than the plan-based approaches. Both the IBL strategy manager and the modularized move realizer are domain and application independent. We have demonstrated the feasibility and flexibility of this approach using a real estate application.

8. References

[1] Barbara J. Grosz, Candace L. Sidner (1986): "Attention, intentions, and the structure of discourse". Computational Linguistics 12(3): 175-204.

[2] Michael F. McTear (1998) "Modeling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit". ICSLP

[3] David Goddeau, Helen Meng, Joe Polifroni, Stephanie Seneff, and Senis Busayapongchai (1996) "A form-based dialogue manager for spoken language applications", pp 701-704, ICSLP.

[4] Barbara Grosz and Sarit Kraus (1996) "Collaborative plans for complex group action." In Artificial Intelligence. 86(2), pp. 269-357.

[5] Jennifer Chu-Carroll, Sandra Carberry (2000): "Conflict resolution in collaborative planning dialogs". *Int*. J. Hum.-Comput. Stud. 53(6): 969-1015.

[6] Karen E. Lochbaum (1998): "A Collaborative Planning Model of Intentional Structure". Computational Linguistics, Volume 24, Issue 4.

[7] Tom Bylander (1991). "Complexity results for planning". In IJCAI, pages 274-279.

[8] D. Chapman (1987). "Planning for conjunctive goals". Artificial Intelligence, 32(3):333—377.

[9] Henry Kautz (1990). "A circumscriptive theory of plan recognition". In Cohen et al. *Intentions in Communication*. MIT Press, Cambridge, Massachusetts

[10] Alicia Abella, Allen L. Gorin (1999): "Construct algebra: analytical dialog management." ACL.

[11] Aha, David W., Breslow, Len A., & Muñoz-Avila, Héctor. (2000). "Conversational case-based reasoning". Applied Intelligence, *14*, 9-32.

[12] Kalyan Moy Gupta (2001), "Taxonomic conversational case-based reasoning", Proc. of Int. Conf. on Case-Based Reasoning: Case-Based Reasoning Research and Development, p.219-233.

[13] Ian Lewin (1998), Autoroute dialogue demonstrator. Technical Report CRC-073, SRI Cambridge.

[14] Claus Zinn, Johanna D. Moore, and Mark G. Core (2002), "A 3-tier planning architecture for managing tutorial dialogue", Proc. of ITS.

[15] David Traum (2002) "Ideas on multi-layer dialogue management for multi-party, multi-conversation, multi-modal Communication (Extended Abstract of Invited Talk)", in Computational Linguistics in the Netherlands 2001: Selected Papers from the Twelth CLIN Meeting.