Asking the Right Questions: Task Hierarchy Predictive Traversal Mechanisms for Mixed Initiative Dialog Management

Juan M. Huerta

IBM T J Watson Research Center, 1101 Kitchawan Road,
Yorktown Heights NY, 10598

Abstract. This paper describes an approach for building conversational applications that dynamically adjust to the user's level of expertise based on the user's responses. In our method, the Dialog Manager interacts with the application user through a mechanism that adjusts the prompts presented to the user based on a hierarchical model of the domain, the recent interaction history, and the known complexity of the domain itself. The goal is to present a conversational modality for experienced or confident users, and a simpler Directed Dialog experience for the more inexperienced users, and to dynamically identify these levels of expertise from the user's utterances. Our method uses a task hierarchy as a representation of the domain and follows a feedback control system framework to traverse of this tree. We illustrate these mechanisms with a simple sample domain based on a car rental application

1 Introduction

Voice-based applications are becoming more prevalent in our lives due mainly to advances in fundamental pattern recognition technology, computer system middleware, and other supporting technologies. Today, we can find voice-based telephony applications aimed at automating call centers, enabling automated transactions for e-commerce and supporting basic customer care management. Whether voice or text based (*e.g.*, [1,7]), conversational technology is emerging as an important type of interface when interacting with enterprise information systems. Examples of architectures implementing telephony-based conversational applications are described in [9,10].

In some domains, simpler Directed Dialog interaction constitutes perhaps a more desirable interaction choice. In a Directed Dialog modality, the application drives the interaction by asking specific questions. Directed Dialog applications are better suited for clearly structured domains, for resolving very specific ambiguities, or for interacting with naïve users. VXML and SALT [8] are examples of typical architectures useful to implementing Directed Dialog applications.

In the Mixed Initiative modality, the application is able to switch between free form interaction (*i.e.*, fully conversational) and directed dialog depending on the needs and expertise of the user and complexity of the active subtask. It is the responsibility of the application (the Dialog Manager, specifically) to determine the nature of the interaction presented to the user at any given time. Examples of mechanisms useful to implementing Mixed Initiative applications are [6, 12].

In this paper we propose a method to dynamically adapt the user interface of a conversational application based on a predictive traversal mechanism of the task hierarchy tree. Combined with this domain hierarchy traversal, we incorporate a mechanism to apply scaffolded prompting in order to allow self-revealing help and prompt rewording. Combined, these approaches enable a system to "Ask the Right Questions" based on the difficulty of the task and the experience of the user, hence driving the interaction at its most natural pace and level of complexity.

This paper is organized as follows: we present a brief background on types of domains or tasks (structured *vs.* unstructured) and discuss their representational difficulties. We then review techniques to represent domains and their prompts, specifically the work previously proposed by Hochberg *et al* describing a method to define structured domains as task hierarchies. We then review the work by Gorin *et al* on Spoken Dialog as a Feedback Control System. We then propose a method to base the traversal of the domain hierarchy on a feedback control mechanism that adapts the modality of the interaction with the user. And finally, we give examples of this technique.

2 Structured and Unstructured Conversational Domains

In certain types of applications, the information that is needed from the user in order to execute a transaction is contained in a clearly defined set of attributes. The relationship between them is also well defined and structured and typically well understood by the application user. Examples of these domains are Travel Reservation and Car Rental. When using voice as the interaction modality, these applications are easily implemented using Directed Dialog and the frameworks that support it (*e.g.*, Voice XML). For the Graphical User Interface it is easy to implement these applications using forms-based markup (*e.g.*, XForms [14]).

It is also possible to implement this type of application with more conversational interfaces. The assumption is that in a single turn of a dialog, a user might include, by his initiative, more than a single token of information, making the overall interaction more efficient. An example of this would be a user spontaneously providing a cluster of information, like all the pickup information pertinent to a car rental transaction.

On the other side of the spectrum are applications where the domain is less structured (*e.g.*, The "*How May I help You"* system [7]) where utterances relate the same information could be worded in a variety of ways and might be embedded in utterances containing additional spurious information. A speech application should be able to deal with these scenarios by incorporating statistical recognizers and parsers that accept less restricted ways of speaking, and represent any extracted information in canonical ways prior to processing it.

3 Tree-based hierarchical representation of Structured Domains

Hochberg *et al* described in [4] a method that uses task hierarchies to provide a basic representation of the domain to the dialog system. The task hierarchy of a domain is a tree in which the leaves are associated with specific canonical values of domain attributes. In the framework proposed by Hochberg *et al*, (called HOT) the root of the hierarchy specifies general methods for managing the dialog, while the intermediate nodes perform actions based on their children or other intermediate nodes. This is a parsimonious representation of the domain that in the HOT framework allows implementing 5 specific relationships of the tasks (*i.e.*, the intermediate nodes) in a hierarchy.

In contrast, other techniques have been proposed to represent a domain, utilizing forms (*e.g.*, Papineni [10]), and schemas, etc. There are also other graph-based representations of semantic domains; for example Wright *et al* [11] and Huerta *et al* [5].

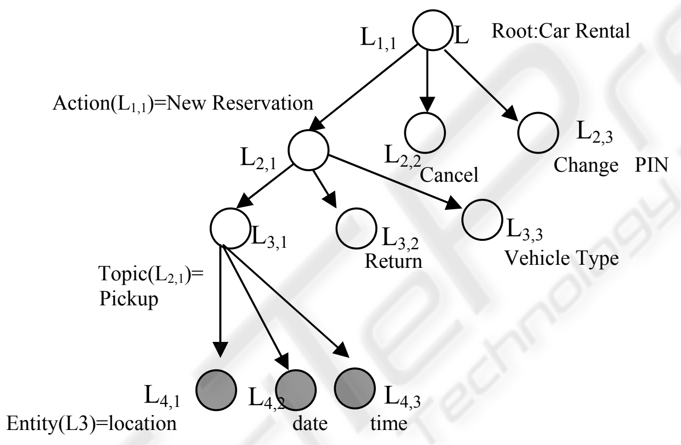

Fig. 1. A partial task hierarchy of a Car Rental application. Colored nodes correspond to terminal nodes.

For the purpose of illustration, Figure 1 shows an example of a task hierarchy for a Car Rental application. For the remainder of the paper we refer to the notation described in this section. The nodes are labeled as $L_{i,j}$ where i is the level of the node and j is its place in its corresponding level. In this example, the terminal nodes are in Level 4 (i=4). In this figure, only the "Pickup" node at level L3 has been extended to terminal nodes. The terminal nodes are represented in colored nodes. In a Speech Recognition based application, such nodes can be recognized and identified by grammars, language models, or name entity detectors. The name of each node contains the value of the attribute named at the first node of each level (*i.e.*, Action is the L2 attribute, Topic is L3, and Entity is L4). Based on the framework employed by the Dialog Manager to traverse the application, the task hierarchy representation is utilized to determine the prompts presented to the user with specific questions. These questions can be triggered by non-terminal nodes (broad questions) or by leaves or terminal nodes (narrow or specific questions). Even at a single node, different wordings can be employed at different times, depending on the situation (*e.g.*, a rewording of the initial prompt might be useful if the user needs help). These mechanisms related to automatic prompt extensions are generally described as prompt scaffolding [13]. In the next subsection we describe how to incorporate prompt scaffolding into the task hierarchy representation.

3.1 Prompt Scaffolding in a Task Hierarchy

When traversing the task hierarchy tree, at any given terminal or non-terminal node a user might need to be presented with prompt rewording, further prompt explanation, or help. Prompt Scaffolding is a common technique in Voice User Interface Design used to introduce successive levels of complexity in help and guidance according to the user's responses. Figure 2 shows and example of scaffolded prompts for a voice interface associated to a connected subset of the nodes of the task hierarchy presented in Figure 1. A prompt $L^{k}_{i,j}$ is associated with the node $L_{i,j}$ in its k^{th} reprompting.

In the same figure, prompt $L^{3}_{3,1}$ corresponds to a template prompt which is utilized when 1 out of the 3 attributes for level $L_{3,1}$ are obtained and a prompt for the remaining 2 is produced. In this prompt `<attname1>` and `<attname2>` are the names of the missing attributes. In a typical Dialog Manager, if there is a required valid input from the user, the system will attempt initial prompting and then subsequent second, third or further prompts until the user generates a valid response. Prompt scaffolding is a simple way to deal with naïve users or difficult spots in an application regardless of interaction modality (Directed Dialog, Mixed Initiative or Conversational).

$L^{1}_{1,1}$: How may I help you?
$L^{2}_{1,1}$: Please choose one of the following new reservation, cancel reservation or change personal information.
$L^{1}_{2,1}$: New Reservation. I need some information about this rental.
$L^{1}_{3,1}$: Please tell me the pickup information.
$L^{2}_{3,1}$: Say a location, date and time for picking up the car. For example "Tomorrow in Boston"'
$L^{3}_{3,1}$: what are the <attname1> and <attname2> for this rental?
$L^{2}_{4,1}$: Please say a pickup location, for example "Boston Airport"?
$L^{1}_{4,1}$: Where do you want to pick up the car?
$L^{3}_{4,1}$: If you know the pickup location say it now, for example "Boston Airport". Or say "transfer" or "help" for assistance.

Fig. 2. Sample scaffolded prompts for a subset of the nodes of the task hierarchy in Figure 1

4 Spoken Dialog Management as a Feedback Control System

A dialog manager (DM) is responsible for driving the interaction with the user in a conversational application. In a mixed initiative system, the dialog manager is responsible for reacting to the user's utterances using an appropriate level of complexity. In other words, the DM's prompts need to be adjusted to the level of sophistication shown by the user. One can frame such interactions as Feedback Control Systems (FCS) in which the DM constantly computes and acts upon a measured *error* or *deviate* between the user's answers and the DM own expectations for such answers. In this section, we describe first the work by Gorin *et al.* [2] on utilizing FCS in speech understanding applications, then we present our proposal for an FCS-based approach to DM.

4.1 Dialog-based interaction as Feedback Control System

In [2] Gorin described a way to frame the interaction between a human and a machine as a feedback control scheme. In his approach, the speech application constantly tracks the likelihood of each action *Ck* in the space of possible actions, given an observed sentence s_l

$$a_k(s_l) = -\log P(C_k \mid s_l)$$

At each point of the dialog, the best action to be taken by the system is determined by the accumulation vector *A*, which is obtained by adding the vector containing the weights (or likelihoods) of the interpretation of the observed sentence *a(si)* with the previous sentence's accumulation's vector *Ai-1* and the error feedback vector $e(s_l)$:

$$A_l = (1-\beta_l)A_{l-1} + \beta_l a(s_l) + e(s_l)$$

The error feedback term in above equation can reflect confidence in the user's input, the user's feedback, focus on specific action parameters, or combine user's and environmental feedback in a multisensory environment, for example [3].

4.2 Mixed-initiative Dialog Management as a FCS

We introduce here a Mixed Initiative based Dialog Manager based on the FCS framework. Our manager aims to identify the state where the application should move to and the interaction modality it should based on the interaction history and *deviates* of the user's input from system's expectations. Figure 3 below depicts or proposed dialog manager organization. The Dialog Manager's functions are split into three boxes: (I) Dialog Manager Semantic Interpretation, (II) DM Comparison operation and (III) DM Parameter Determination & Prompt Generation. Variables of this system are: *action*, *context node*, expected user's proficiency, *observed user proficiency, input*, *prompt*, and *control differential*. These variables are explained below.

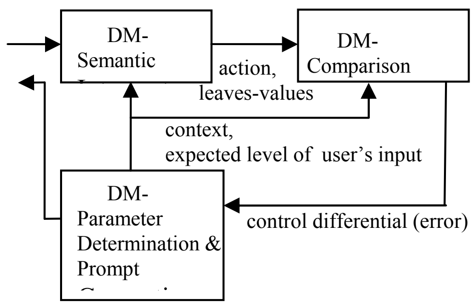

Fig. 3. Block diagram of an FCS-based Mixed Initiative Dialog Manager

The process starts when the system observes the user's input and based on the existing current context it executes a specific action. We assume that the system maintains at each moment an expected level of user performance based on previously obtained user responses. The DM executes a comparison between this expected level of response and the actual observed response and computes the difference of these two parameters (we assume that the system is able to map the complexity of a user's utterance into a numeric quantity). The difference or deviate is sent to the DM-Parameter Determination prompt, which generates a new prompt and establishes a new context depending on the predicted level of proficiency of the user after determining which attribute or set of attributes should be obtained next from the user. Ideally the DM will toggle between conversational and directed dialog modalities based on the predicted user's performance.

In contrast to the approach described by Gorin in [2], our approach does not try to directly find a most-likely action based on the speaker's utterance and topic or action likelihoods. Instead, the DM tries to populate the attributes of the domain's data model and assign them values in an efficient way. To do this the system constantly tracks the context of the dialog and establishes the best modality of interaction. Also, in contrast to Gorin's approach, this system does not follow a probabilistic (or maximum likelihood) formulation: the values of the system's variables are measurements of the observations and no probabilistic model is assumed. This is true not only for the DM-Semantic Interpretation block, but also for the DM-Parameter Determination & Prompt Generation block. In the next section we will tie this basic idea to a Domain Hierarchy model of the task.

5 Domain Hierarchy Predictive Traversal for Mixed Initiative

In essence, the goal of a domain hierarchy-based Dialog Manager is to aid the traversal process of the domain hierarchy tree. When sufficient terminal nodes to execute a query or transaction are collected, the DM triggers the appropriate query or transaction. In a directed dialog application, the tree is traversed as a predefined sequence of terminal nodes (for example, in VXML the FIA (form interpretation algorithm) is responsible to traverse the forms of the application and does so in a deterministic way). The intermediate nodes serve to establish a hierarchy of leaves (*e.g.*, in a VXML document the terminal nodes are the fields and the intermediate nodes the forms). In a mixed initiative task the dialog manager can be thought of traversing the tree spanning both terminal nodes and non-terminal nodes depending on the user's input, allowing in this way for the collection of multiple tokens through a single utterance (*i.e.*, the user might take the initiative to include in its input more that one token, or information not pertaining to the active node). We explain here how a dialog manager can promote higher levels of interaction by avoiding terminal nodes when a certain level of expertise or control by the user is inferred in a FCS type of framework.

5.1 FCS based Predictive Tree traversal Algorithm

We now describe a simple method that incorporates the concept of FCS applied to dialog management described in section 4.2 with the task hierarchy DM discussed in section 5.1. In essence, the purpose of the DM, as we have mentioned, is to both present the context (that in our case is associated with a node in the domain tree) and determine the prompt to be presented to the user. The resulting utterance will be used to: (a) populate pertinent leaves of the tree and (b) judge the level of proficiency of the user in the application updating the context and the interaction level with the user.

This process is iterated until a sufficient set of leaves are populated. We now present a basic algorithm which implements a DM that operates on a domain tree, and based on a FCS framework traverses the domain in an adaptive way toggling between Directed Dialog and Mixed Initiative:

```
1- Start: Initiate the context to be root node. Assume an initial
   neutral DM policy.
2- Play prompt based on context.
3- Receive user's input. Perform DM-Semantic interpretation and
   populate relevant leaves.
4- Perform DM-comparison of expected vs. observed user's profi-
   ciency, adjust the deviate parameter.
5- DM-Parameter determination: Adjust DM-policy, update context.
6- If a sufficient set of leaves is populated then stop, otherwise
   go to 2.
```

The algorithm above is a simple implementation of the process taking place in figure 3. Because the first prompt is always the same, the initial DM strategy is set to be "neutral" which is not a third DM policy, but instead, denotes the undefined state of the DM policy. After the first user's utterance is observed, the DM can establish the proficiency and move to determine whether directed dialog or mixed initiative should be used. The value of deviate parameter can be set according to the following policy:

```
if  observed_proficiency > expected_proficiency
     then deviate = +1
else,
if observed_proficiency = expected_proficiency
      then deviate = 0
else
     deviate = -1
end.
```

Therefore, instead of allowing state variable *deviate*, which tracks the error, to have a continuous value, it is only permitted to have 3 values: {+1,0,-1}, which loosely correspond to: better than expected, expected and worse than expected answers, respectivelly.

The policy to determine whether the observed proficiency is larger, equal to or smaller than the user's proficiency is described in the following policy table:

This condition is true	**When the following situation is observed:**
`observed_proficiency > expected_proficiency`	The system is in Directed Dialog and system capbures two or more leaf (terminal nodes) values
`observed_proficiency = expected_proficiency`	The system is in Directed Dialog and there is only one leaf filled, or, The system is in Mixed Initiative and all prompted leaves are filled
`observed_proficiency < expected_proficiency`	The number of leaves filled by the DM-Semantic Interpretation block under the prompted context is either one, zero or less than the number of leafs prompted in last interaction

The algorithm described in this section can be combined with a prompt scaffolding strategy of the type discussed previously. Similarly to the Dialog Manager, a Prompt Manager strategy can be incorporated into the FCS algorithm describe above. To do this an additional error variable that tracks the response to various levels of verbosity can be added. For simplicity, we avoid here introducing such variable.

6 Examples

In this section we provide examples of the FCS-based Predictive Tree traversal algorithm described in Section 5.1. We illustrate the parameter value setting portion of the algorithm in the context of dialog interactions. We refer to the Car Rental domain illustrated in Figures 1 and 2. Table 1 and 2 below show interactions between a User (U) and a System (S) that implements the algorithms described in section 5.1, and on the Car Reservation subdomain introduced previously in this paper. Column 1 shows each turn's system prompts and user response and the other columns show the values the dialog manager components assign to system variables.

Table 1 illustrates the scenario where the user is generally proficient in the domain and traverses the domain primarily in mixed initiative modality. At the third response, however, the user returns less than then expected number of attributes and the *deviate* variable is set to -1. This switches the conversation into Directed Dialog which produces a fourth dialog turn query specific for the missing attribute. Table 2, illustrates an interaction where the user mostly interacts in directed dialog. In the last turn the user provides a multiple-attribute utterance, allowing for the completion of the query in that utterance. Not described is the execution of the transaction.

Prompt/Response	Context Leaf	Semantic Interpretation:	DM System Parameters
S*: Welcome. How may I help you?*	context=$L^{1}_{1,1}$		

U: I'd like to rent a car in Pittsburgh's airport		action=rent
		pckp_loca=PGH
			modality= -
			deviate=+1

S: *What are the pickup date and pickup time for this rental?*	context= $L^{3}_{3,1}$:		

U: Tomorrow at noon.		pckp_date=today+1
		pckp_time=noon
			modality= MI
			deviate=0

S: *Please tell me the return information?.*	context= $L^{1}_{4,2}$		

U: Same location, the morning after		ret_location=PGH
		ret_date =today+2
			modality= MI
			deviate=-1

S: *What is the return time for this rental?*	context= $L^{1}_{4,1}$		

U: Ten A. M.		return_loca=PGH	modality= DD
			deviate=0

S: *I will reserve a car for tomorrow noon in Pittsburgh International Airport*	context= completed_tree		

Table 1. Example of a interaction with a Predictive Tree Traversal-based DM

Prompt/Response	Context Leaf	SI.	DM Params.
S: *Welcome. How may I help you?*	context=$L^{1}_{1,1}$		

U: [silence]			modality= -
			deviate=+1

S: *Please choose one of the following: new reservation, cancel, or information*	context= $L^{2}_{1,0}$		

U: New Reservation.		action=new	modality= DD
			deviate=0

S*: New Reservation. Let me ask you some information about this reservation*	context= $L^{1}_{3,2}$		

U: It's for Pittsburgh international airport		pckp_loca=PGH	modality= DD
			deviate=-0

S: *What are the pickup date and pickup time for this rental?*	context= $L^{3}_{4,1}$		

U: Tomorrow noon. And the return is the day after, same location, at noon.		pckup_time=noon
		ret_loca=PGH
		ret_date=today+2
		ret_time=noon
			modality= DD
			deviate=+1

S: *I will reserve a car for tomorrow noon M in Pittsburgh International Airport*	context= completed_tree		

Table 2. Example of a interatcion with a Predictive Tree Traversal-based DM

Conclusions

We presented in this paper a novel algorithm intended to implement a Mixed Initiative Dialog Manager for conversational applications. Our technique uses the task hierarchy description of a conversational domain. Prompts for the Directed Dialog modality are incorporated into the terminal nodes of this hierarchy. Prompt scaffolding, aimed to support self revealing help, can be easily integrated into this framework. Thus we demonstrated that a tree representation of the domain is a versatile representation as it easily supports mixed initiative and directed dialog.
Furthermore, we presented a mechanism that based on the task hierarchy representations implements an *adaptive* Mixed Initiative interaction. A Mixed initiative dialog is the result of simultaneously enabling Directed Dialog and Multi-token interaction. Based on a feedback control framework our method predicts the interaction modality that best fits the user abilities in every given part of a transaction based on the immediate history of the dialog and other parameters of the interaction. We illustrated this mechanism with a basic car rental application. The combination of a predictive framework (FCS) and a tree representation of the hierarchical domain constitute a concise and flexible framework for developing Mixed Initiative systems.

References

1. AT&T "Need Help? Ask Allie ™" Web Application.
2. Gorin, A. L., ``Spoken dialog as a feedback control system,'' Proc. ESCA Workshop on spoken dialog systems, Visgo, Denmark, May 1995, pp. 173-176
3. Gorin, A. L., Levinson, S. E. and Sankar, A., ``An experiment in spoken language acquisition,'' IEEE Trans. Speech and Audio, vol. 2, no. 1, Part 2, Jan. 1994, pp. 224-240.
4. Hochberg J. , Kambhatla N., Roukos Salim, "A Flexible Framework for Developing Mixed-Initiative Dialog Systems", 3rd Sigdial Workshop on Discourse and Dialogue, 2002
5. Huerta "Graph based representations and techniques for NLU Application Development" Proc. ICASSP 2003, Honk Kong.
6. Louwerse, M. Graesser A. Olney A. "Good Computational Manners", AAAI Symposium Technical Report, November 15-17, 2002, North Falmouth, Massachusetts
7. A.L. Gorin, G. Riccardi and J.H. Wright How May I Help you Speech Communication, vol. 23, pp. 113-127, 1997
8. W3C, VXML 2.0 http://www.w3.org/TR/voicexml20/
9. The MIT Galaxy System http://www.sls.csail.mit.edu/GALAXY.html
10. Luo X., Papineni K, "IBM Darpa Communicator V 1.0", Darpa Communicator Principal Investigators Meeting, Philadelphia USA 2000.
11. Wright Jerry, Allen Gorin, Alicia Abella "Spoken Language Understanding within Dialogs Using a Graphical Model of Task Structure", Proc. ICSLP, Sydney 1998.
12. Meng, Wai and Pieraccinni, "The Use of Belief Networks for Mixed Initiative dialog Modeling" ICSLP 2000, Beijing China.
13. Guzdial, M. Software-realized Scaffolding to Facilitate Programming for Science Learning. Interactive Learning Environments, Vol. 4, No. 1, 1995, 1-44.
14. W3C, "XForms" http://www.w3.org/MarkUp/Forms/