Dynamic User Level and Utility Measurement for Adaptive Dialog in a Help-Desk System

Preetam Maloor
Department of Computer Science,
Texas A & M University,
College Station, TX 77843, USA
preetam@csdl.tamu.edu

Joyce Chai
Conversational Machines
IBM T. J. Watson Research Center,
Hawthorne, NY 10532, USA
jchai@us.ibm.com

Abstract

The learning and self-adaptive capability in dialog systems has become increasingly important with the advances in a wide range of applications. For any application, particularly the one dealing with a technical domain, the system should pay attention to not only the user experience level and dialog goals, but more importantly, the mechanism to adapt the system behavior to the evolving state of the user. This paper describes a methodology that first identifies the user experience level and utility metrics of the goal and sub-goals, then automatically adjusts those parameters based on discourse history and thus directs adaptive dialog management.

Introduction

A new generation of dialog systems should be viewed as learning systems rather than static models (Jokinen, 2000). Close-world and static approaches have tremendous limitations and often fail when the task becomes complex and the application environment and knowledge changes. Thus, the learning capability of a dialog system has become an important issue. It has been addressed in many different aspects including dynamic construction of mutual knowledge (Andersen et al, 1999), learning of speech acts (Stolcker et al, 1998), learning optimal strategies (Litman et al, 1998; Litman et al, 1999; Walker et al, 1998), collaborative agent in plan recognition (Lesh et al, 1999), etc. This paper addresses the dynamic user modeling and dialog-goal utility measurement to facilitate adaptive dialog behavior.

For any dialog system dealing with a technical domain, such as repair support (Weis, 1997), help-desk support, etc, it is crucial for the system not only to pay attention to the user knowledge and experience level and dialog goals, but more important, to have certain mechanisms that adapt the system behavior in terms of action planning, content selection, and content realization to user cognitive limitations. Dialog strategies and management should be adjusted to the evolving state of the user. Thus a better understanding and modeling of user cognitive process and human perception is desirable.

In this paper, we propose a methodology that automatically learns user experience levels based on sub-goal utilities and characteristics observed during the interaction. Those user levels will further feedback to update utility metrics and direct different dialog strategies at each level of dialog management: action planning, content selection and content realization. The Help-Desk is our application domain. This is a work in progress. We have built a prototype system and are currently in the process of evaluation of our methodology and hypotheses.

1 System Overview

The system components, shown in figure 1, consist of a problem space representation and a set of modules and agents that utilize this representation. The architecture supports a dynamic updating process for user level and sub-goal utility measurement, and thus allows the system to adapt its dialog behavior to the updated environment.

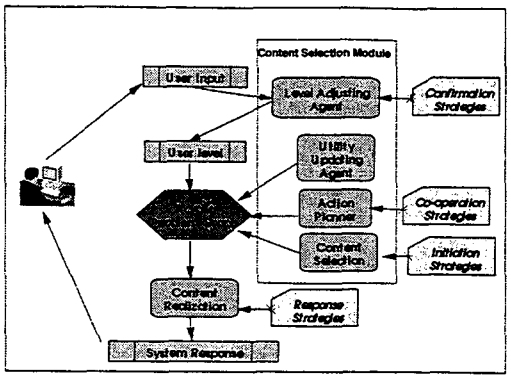

Figure 1. System Components

The problem space is modeled by an Acyclic Problem Graph structure, which represents the dialog goal (i.e., final goal) and different paths (solutions) to the final goal. The Level Adjusting Agent controls the initial detection and dynamic shifting of user expertise level based on the interactions with the user. The Action Planner identifies the problem node (i.e., dialog goal) in the Acyclic Problem Graph and locates the optimal path to it. The Content Selection component uses the Level Adjusting Agent and the Action Planner to select the content for the dialog. The Content Realization module deals with the final presentation of the dialog content to the user. The Utility Updating Agent automatically updates the utility metrics of the sub-goals in the Acyclic Problem Graph based on the single and group user models that are created during interactions. Different strategies are applied in different modules, which will be described later.

2 Problem Space Modeling

The problem space is modeled by an acyclic graph named Acyclic Problem Graph. It can also be considered as a forest containing joint trees that have overlapped root nodes and internal nodes. Internal nodes correspond to sub-goals. A path traversed from a root to a particular node contains a potential solution to a goal or sub-goal related to that node. Given a root node, the further away from the root, the greater is the complexity of the goal (or sub-goal) represented by a node. Since multiple paths can lead to a node, there could be multiple solutions to a goal.

Figure 2 is a fragment of an acyclic graph for solving problems pertaining to a Windows based PC. In this example, three paths correspond to three potential solutions to the problem about how to set the display resolution of a monitor.

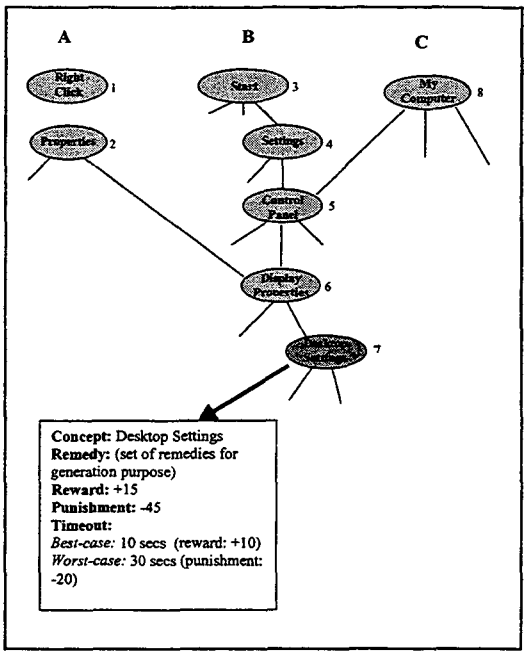

Figure 2. Acyclic Problem Graph

Each node in the graph has the following fields: Concept Name, Remedy, and Utility Metrics that include Reward, Punishment, Best-case timeout and Worst-case timeout.

Concept Name represents an instruction corresponding to a particular goal or sub-goal during the problem solving. For example, the concept of "Display Properties" node deals with manipulating the display of the monitor.

Remedy is the template that is used to generate natural language responses and explanations corresponding to a particular goal. It also contains phrases and key terms used for language generation.

Reward and Punishment are the utility metrics corresponding to each sub-goal (Winkler,

1972) depending upon the hypothesis of *uncertainty of understanding* and the *level of importance*. Uncertainty of understanding implies the difficulty in following certain instructions or understanding certain concepts. For example, some technical terms require users to possess greater expertise in order to comprehend them. Some potential ways of initializing uncertainty of understanding are by observation, analysis of previously logged data, or surveys. The level of importance indicates the importance of the sub-goal for understanding an instruction or a concept towards the realization of the overall goal of solving the problem. One good indication of such importance, for example, in the Acyclic Problem Graph, is the branch factor of each node. A more difficult concept has a greater level of uncertainty and hence would lead to less punishment if the user does not understand it. On the other hand, if a user correctly understands a concept that has a high degree of uncertainty, he would be rewarded highly. Reward and punishment can be pre-determined and then re-adjusted later when the user and the group modeling progresses.

Timeout metrics are used to indicate whether the user understands the instruction or the concept associated with the sub-goal within the expected period of time. The hypothesis is that when a user has no problem of understanding a system instruction, the user is very likely to respond to the system rapidly. However, when the user has difficulties, he/she tends to spend more time on thinking and asking for help. There are two timeouts: *best-case and worst-case.* Each timeout has a reward and a punishment. Best-case time is the time expected by the system, in the best case, that a user would take to understand the instruction. The user is rewarded when actual time spent is less than the best-case time. Similarly, the worst-case time is the system expectation for the worst case. If the user still doesn't get the instruction after the worst-case time period, he is punished for it. Again, these values are pre-set and will be dynamically re-adjusted.

3 Dialog Management

The Dialog Manager can be broadly classified into two main modules: Content Selection and Content Realization.

3.1 Content Selection Module

The Content Selection Module consists of four components: Level-Adjusting Agent, Utility-Updating Agent, Action Planner and Content Selector.

3.1.1 The Level-Adjusting Agent

There are three levels of user expertise that the dialog manager takes into consideration: Expert, Moderate and Novice. The agent controls the initial detection and dynamic shifting of user expertise level based on interactions with the user.

If a user is using the system for the first time, a good indication of the initial user expertise level is the level of detail and technical complexity of the initial query. As user's interaction with the system continues, a profile of the user is constructed gradually. This profile could be re-used to set the initial user expertise when the user uses the system again.

The dynamic shifting of user expertise level is of two kinds: local (i.e., temporary) shifting between local expertise levels and accumulated (i.e., long term) shifting between accumulated expertise levels. Local shifting adjusts the expertise level temporarily - by observing the user confirmation (currently an explicit user confirmation is expected) which indicates whether he/she understands a certain instruction. The reason for temporary adjustment is because we assume that the user is having trouble understanding only this particular instruction and not the overall solution.

The accumulated shifting permanently adjusts the user expertise level depending upon two threshold values: EXPERT_LEVEL and NOVICE_LEVEL. The user is considered an expert when his *accumulated* expertise level is above the EXPERT_LEVEL and is considered

novice when that is below the NOVICE_LEVEL. The user is assumed to have moderate expertise if he lies between these two thresholds. An accumulated value (ACCUM_VALUE) is calculated based on the whole dialog history. If the ACCUM_VALUE of a user crosses a threshold, the *accumulated* user expertise level changes long term as it is assumed that there is a change in the user's overall understanding of the solution.

At any point of the interaction, the system maintains ACCUM_VALUE for the user. This value is used to adjust the user expertise level. The ACCUM _VALUE is calculated based on the following set of features associated with utility metrics in each node in the discourse history (Wies, 1997; Jameson et al, 1999):

Sub-goal Complexity: More complex sub-goals have a greater level of importance and uncertainty of understanding, and thus have a high reward and a low punishment. Similarly, comparably simple sub-goals have a low reward and a high punishment.

Accomplishing Time: this is perhaps the trickiest parameter as the chance of making an incorrect assumption is much higher. The user response time could be a good indication of user understanding. The longer the resolving of the user's problems lasts, the more unfavorable the evaluation is. Also if the user responds quickly, he is rewarded for it. To detect whether the user is distracted or not, if a series of timeouts occur continuously, the user is not paying attention to the system.

Response Complexity: There is a reward and a punishment associated with each system response that reflects the complexity of the content and realization of the system responses. First of all, the content for response generation varies with different expertise levels. For novice users, all the content on the solution path will be generated as several turns of responses based on the number of sub-goals in the path. For expert users, only 40% content on the solution path (toward the final goal) is used for the generation as one response. Furthermore, for users with different expertise level, the Content Realization Module will generate system responses (in the prototype system, the system responses are mainly instructions that guide users to solve a help-desk problem) with different levels of syntactic complexity and technical details. For example, for novice users, the system tends to generate responses with single instruction corresponding to one sub-goal, while for expert users, the system tends to generate responses with single instruction corresponding to multiple sub-goals on the solution path. The response with multiple instructions will have higher reward and lower punishment than those are associated with single instruction. Thus the user who gives a positive confirmation to a more complex system response will be rewarded higher than those who understand a simple system response.

Based on the above factors, the ACCUM_VALUE can be calculated depending upon the conditions using the following formulae:

ACCUM_VALUE = ACCUM_VALUE + *f*[response -complexity (reward, punishment), sub-goal(reward, punishment) , timeout(reward, punishment)]

In the prototype system, we have used the following:

If a goal is accomplished by the user(indicated by positive user confirmation),
ACCUM_VALUE = ACCUM_VALUE + [response-complexity(reward) * sub-goal(reward)]

If a goal is not accomplished(indicated by negative user confirmation),
ACCUM_VALUE = ACCUM_VALUE - [response-complexity(punishment) * sub-goal(punishment)]

If a goal is accomplished before *best-time* timeout value,
ACCUM_VALUE = ACCUM_VALUE + [response-complexity(reward) * sub-goal(best-case timeout reward)].

If a goal is not accomplished before *worst-time* timeout value,
ACCUM_VALUE = ACCUM_VALUE - [response-complexity(punishment) * sub-goal(worst-case timeout punishment)].

Other variations of the formula are expected to be explored in the future.

3.1.2 *Action Planner and Content Selector*

The Action Planner identifies the final goal node in the Acyclic Problem Graph and finds the optimal path to it. The optimal path is selected based on the *path utility* function. The *utility* of a path in the graph is the summation of the reward/punishment ratio of all the nodes (sub-goals) in that path.

$$\textit{Path utility (start-node, goal)} = \frac{\Sigma (r_i / p_i)}{n}$$

where i is a concept node in the path from the start node to the goal node, r_i is the reward and p_i is the punishment of the corresponding node i. The number of nodes n in the path acts as the normalizing factor.

Thus for a given path, higher its *path utility,* greater is the difficulty to understand the concepts it contains and thus higher is the level of expertise required.

The following co-operative strategies are used: for an expert user, select the path that has the maximum *path utility*. For a novice, select the one with the minimum *path utility* since this is the one containing concepts easiest to understand and with more steps of instructions. For a moderate-experience user, select a path in between. (We are currently more focused on the experienced and novice users.) Content Selector is applied to select the appropriate nodes on the path to form the content of dialog.

3.1.3 *Utility Updating Agent*

A set of users having very similar expertise levels can be classified as a group. A *Utility Updating Agent* dynamically updates utility metrics of sub-goals in the Acyclic Problem Graph based on the group interactions with the system. For example, Group A has a reward of +50 and a punishment of –10 assigned to the sub-goal with associated concept of *Display Properties*. However the agent notices that the majority of the group understand the corresponding instruction very quickly without going into the sub-goal resolution, then the agent decreases the reward to +35 and increases the punishment to –25. This dynamic re-training of utility metrics in sub-goals would reflect the evolving user experience level as a whole and would improve the robustness of the dialog manager.

3.2 Content Realization Module

This module deals with the final presentation of the dialog content to the user. The dialog manager adopts different response strategies for each of the three expertise levels. It has been observed that an *expert* user appreciates a response, which is precise, to the point, and short. For a *novice* user, it has been observed that such a user likes system instructions that are step-wise, higher level of detail and minimum technical jargon. For a moderate-experience user, the strategy lies somewhere in between which we haven't given a full consideration. The response strategy followed for each type of user is given in the table 1.

Response / Expertise	Level of detail of system instructions and explanation	Technical terms in system instructions and explanation	Syntactic Conciseness of the explanation
Expert	Low	High	High
Moderate	Moderate	Moderate	Moderate
Novice	High	Low	Low

Table 1. User expertise level and corresponding dialog strategies

3.3 Algorithm

The proposed algorithm for action planning, content selection and content realization is given in Figure 3. This algorithm recursively applies a divide and conquer mechanism to accomplish the final goal by resolving sub-goals. Two variables (i.e., local expertise level and accumulated expertise level) are maintained by the Level-Adjusting Agent for the automated level updating. The Action Planner identifies the goal node and the solution path to it depending on the expertise level of the user. Based on this level, the Content Realization Module will first select the content on the path to be presented and then use various response strategies to generate and display system instructions to the user. For novice users, all the content on solution path will be used; for moderate and expert users, only partial content on the path (toward the goal) will be used. In terms of generation, for novice and moderate expertise users, the system generates responses with single instruction corresponding to one sub-goal, while for expert users, the system tends to generate responses with single instruction corresponding to multiple sub-goals on the solution path. The syntax of the response becomes more complex as the expertise level increases. Depending on the response of the user, the Level-Adjusting Agent updates the user expertise level and adapts the response strategies accordingly.

1) Level-Adjusting Agent detects the initial expertise level and assigns it to both local expertise level and accumulated expertise level.
2) Action Planner identifies the start node and goal node in the Acyclic Problem Graph and locates the appropriate path between the start node and the goal node.
   a. For novice user, the path with minimum path utility is selected
   b. For expert user, the path with maximum path utility is selected
   c. For moderate user, a path in between is selected
3) Content Realization Module generates system instructions based on the selected path by using the following response strategies:
   a. For an expert, the instruction is generated by using the nodes that fall within a distance of X% from the goal node to the root node.
   b. For a moderate-experienced user, nodes within a distance of Y% (where Y > X) are used.
   c. For a novice, all nodes from the root to the goal are used to generate the instruction
   (X and Y could be experimentally determined later)
4) Content Realization Module displays generated instructions to the user.
5) Level-Adjusting Agent receives the user confirmation and updates user expertise level.
   a. If the confirmation is positive, the Level Adjusting Agent does the following:
      i. Update ACCUM_VALUE=ACCUM_VALUE + [response-complexity(reward) * sub-goal(reward)]
      ii. If ACCUM_VALUE crosses above an expertise level threshold, upgrade *accumulated* expertise level
      iii. If the goal node is the final node, exit. Otherwise, continue to the next node.
   b. If the confirmation is negative
      i. If current *local* expertise level is greater than *novice,* temporarily reduce local expertise level; else suspend system at current state (so that the user can take his own time in understanding the instruction or seek outside help).
      ii. Update ACCUM_VALUE= ACCUM_VALUE – [response-complexity(punishment) * sub-goal(punishment)].
      iii. If ACCUM_VALUE crosses below a level threshold, reduce *accumulated* experience level.
      iv. Record the current node and the current path
      v. Make current node as the goal node; Go to step 2. Repeat until all sub-goal nodes of this goal node are understood.
6) Re-initialize *local* expertise level to current value of *accumulated* expertise level. Restore path to value stored in step 5.b.iv. Go to step 2. Reset the start node. Continue till the final goal is reached.

(A timer that is running on a separate thread also modifies the ACCUM_VALUE variable. On occurrence of a timeout, the following steps are followed:
If the time spent is less than the *best-case time*
ACCUM_VALUE=ACCUM_VALUE + [response-complexity(reward) * sub-goal(best-case timeout reward)].
Go to step 5.a.ii.
If the time spent is more than the *worst-case time*
ACCUM_VALUE=ACCUM_VALUE – [response-complexity(punishment) * sub-goal(worst-case timeout punishment)].
Go to step 5.b.iii.)

Figure 3. Algorithm

4 An example of interaction

A scenario application for the system using the problem tree of Figure 2 is given in Figure 4. This set of interactions is between the system and a user who is trying to adjust the display settings of his monitor.

1. System: Hi, how may I help you?
2. User: I need to adjust the desktop resolution of my monitor
- Initial *accumulated* expertise level detected – Expert
- Goal node –Desktop Setting (as it deals with adjusting the resolution)
- Shortest part is nodes 1-2-6-7 (assuming that it has the maximum *path utility* value).
- Nodes 6,7 (40% rounded to nearest whole number) are selected to generate response.
- Response generated with instruction corresponding to multiple sub-goals on the solution path.

3. System: You can do that by adjusting the desktop area in the desktop settings section of the display properties window.
4. User: Where is desktop setting window?
- Temporarily decreasing *local* expertise level to Moderate.
- ACCUM_VALUE=ACCUM_VALUE - [response -complexity (punishment) * Desktop-Setting(punishment)]
- Shortest path to Desktop Setting recalculated as 8-5-6-7 (assuming that it has the average *path utility* value).
- Nodes 5,6,7 (70% rounded to nearest whole number) are selected to generate response.
- Response generated with single instruction corresponding to one sub-goal.

5. System: First, go to the Control Panel window.
...worst-case timeout occurred....
- Temporarily decrease *local expertise* level to Novice.
- ACCUM_VALUE=ACCUM_VALUE - [response-complexity(punishment) * control-panel(worst-case timeout punishment)]
- ACCUM_VALUE falls below EXPERT_LEVEL. Make *accumulated* experience level moderate.
- Shortest path to Control Panel recalculated as nodes 3-4-5 (assuming it has the minimum *path utility* value).
- All nodes used to generate response.
- Response generated with single instruction corresponding to one sub-goal.

6. System: 1. Click on "Start" icon on lower left corner of your screen.
7. User: Ok
- ACCUM_VALUE = ACCUM_LEVEL + [response-complexity (reward), Start(reward)]

8. System: 2. Click on "Settings" label.
9. User: Ok
- ACCUM_VALUE=ACCUM_LEVEL + [response-complexity(reward), Settings(reward)]

10. System: 3. Click on "Control Panel" label.
- No change in ACCUM_VALUE as user has been punished earlier at this step.
- *local* experience level goes back to moderate. Path reset to 8-5-6-7 selected in step 4.

11. System: Now click on the Display Properties window.
12. User: OK
- No change in ACCUM_VALUE as user has been punished earlier at this step.

13. System: Select "Desktop settings" in this window.
14. User: Ok
- No change in ACCUM_VALUE as user has been punished earlier at this step.

15. System: Finally, adjust "display area" by moving the slide-bar.
16. User: Ok. Thank you
- No change in ACCUM_VALUE as user has been punished earlier at this step.
- Profile of user stored.

Figure 4. An example

Notice that after step 9, the user level remains at moderate and does not return to expert. This is because a decrease in *accumulated* user expertise level has taken place in step 5. Also, the style of the dialog is different at each level according to the expertise level of the user at that time.

Conclusion

This paper describes hypotheses, strategies and a methodology in building a dialog system that adapts itself to the evolving level of user expertise. Furthermore, the system allows automated sub-goal utility adjustment based on history of interactions with groups of users. We have implemented the algorithm described in this paper on a prototype system where the utility metrics have been initialized manually by a help-desk expert, based on his experiences of interaction with users. We are currently working on evaluation of hypotheses and the system.

This work is still in its early stage. Our future work includes conducting evaluation of the hypotheses and the system and investigating machine learning techniques for improving utility adjustments.

Acknowledgement

This work was a summer project while the first author was doing his summer internship at the Conversational Machines Group at IBM T. J. Watson Research Center. We would like to thank all members in Conversational Machines Group for their discussions and support.

References

Carl Andersen, David Traum, K. Purang Darsana Purushothaman, Don Perlis (1999) *Mixed Initiative Dialogue and Intelligence via Active Logic*. In proceedings of the AAAI'99 Workshop on Mixed-Initiative Intelligence, pp. 60-67.

Anthony Jameson, Ralph Schäfer, Thomas Weis, André Berthold and Thomas Weyrath (1999) *Making Systems Sensitive to the User's Time and Working Memory Constraints*, Intelligent User Interfaces.

Kristiina Jokinen (2000) *Learning Dialog System.* LREC 2000 Second International Conference on Language Resources and Evaluation, Athens, Greece.

Neal Lesh, Charles Rich, Candace Sidner (1997) *Using plan recognition in human-computer collaboration.* In 7th International Conf. On User Modeling, Banff, Canada.

Diane J. Litman, Shimei Pan, Marilyn A Walker, (1998) *Evaluating Response Strategies in a Web-Based Spoken Dialogue Agent.* In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98), Montreal, Canada, pp. 780-786.

Diane J. Litman, Shimei Pan (1999) *Empirically Evaluating an Adaptable Spoken Dialogue System.* In Proceedings of the 7th International Conference on User Modeling (UM), Banff, Canada, pp. 55-64.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Noah Coccaro, Daniel Jurafsky, Rachel Martin, Marie Meteer, Klaus Ries, Paul Taylor, Carol Van Ess-Dykema (1998) *Dialog act modeling for conversational speech.* In Chu-Carroll J., and Green N., (Eds), Applying Machine Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium. Stanford, CA.

Marilyn Walker, Jeanne Fromer, Shrikanth Narayanan (1998) *Learning Optimal Dialog Strategies: A Case Study of a Spoken Dialog Agent for Email.* In Proceedings of COLING-ACL'98, University of Montreal, Canada.

Thomas Weis (1997) *Resource-Adaptive Action Planning in a Dialogue System for Repair Support,* KI.

Robert L Winkler (1972) *Introduction to Bayesian Inference and Decision.* Holt, Rinehart and Winston Inc.