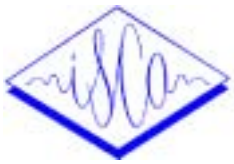
ISCA Archive
http://www.isca-speech.org/archive



COST278 and ISCA Tutorial and Research Workshop
on Robustness Issues in Conversational Interaction
University of East Anglia, Norwich, UK
August 30-31, 2004

Dynamic Choice of Robust Strategies in Dialogue Management

Bryan McEleney and Gregory O'Hare

Department of Computer Science,
University College Dublin, Ireland

bryan.j.mceleney@ucd.ie, gregory.o'hare@ucd.ie

Abstract

An important tradeoff in error-prone dialogue is between the cost of using more robust dialogue strategies and the cost of recovering from failed understanding without using them. A strategy has to be quantitatively planned for each dialogue state, since too robust a strategy might not have a worthwhile effect on the failure rate. A dialogue manager is described which chooses between strategies that have differing levels of robustness with a view to maximising the efficiency of the dialogue.

1. Introduction

A common measure for the performance of a dialogue system is the time-to-completion of a given task [1]. The time taken is influenced by whether a robust dialogue strategy is chosen. For example, where the system is trying to recognise the hearer's intention and cannot produce a single definite hypothesis, the agent can use a confirmation subdialogue. Doing so might increase the time-to-completion since it requires effort from both the system and the user. On the other hand, using a confirmation reduces the likelihood of an error in recognising the user, entailing less chance of a costly correction subdialogue later on. Similarly, in generating system utterances, there is some choice available about whether to use ambiguous forms such as using anaphora and ellipsis, or words or phrases which while brief, are not robust since they risk not being informative enough about the system's intention. Other strategy choices can be made, such as whether the speech synthesis system should speak quickly or slowly, or whether a prosodic cue would be useful to resolve a potential ambiguity.

Within the SAID project, an agent-based approach is taken to development of dialogue systems, including a distributed, robust speech recognition system [2], and a robust agent-based dialogue management system. The dialogue management system is based on Rao and Georgeff's BDI [3] (belief-desire-intention) theory of agent decision-making. The system, whose design is presented in this paper, has been implemented in PROLOG and tested on various example problems such as planning confirmations, and deciding between robust and less robust utterance generation strategies. It is a domain independent system, only requiring instantiation of a plan-rules file for the dialogue task. Examples of the system's operation are given in this paper, and it is shown that probabilistic plan recognition and adaptation of a user model are both important in deciding dialogue strategies in these examples.

2. Dialogue Planning Model

Rao and Georgeff's [3] theory of agent choice is based on a time-tree representation of possible dialogue outcomes. Similar to a decision tree, there are choice nodes that represent the strategies the agent may use at his turn in the dialogue. There are also chance nodes that represent the uncertainty of the agent's environment, which in the context of a dialogue system, can be used to represent uncertainty in the beliefs or intentions of the agent acting at the following choice node. The time-tree represents the alternating actions of the agents, with each agent's turn represented by a choice node in the shared multi-agent plan.

A nested belief model (user-model) is employed by the system to represent the system's beliefs about the user's mental state (ie. beliefs and intentions), the system's beliefs about the user's beliefs about the system' mental state and so on. A deeply nested model is necessary, since in system generation of utterances, the system expects the user to interpret the utterance according to the user's model of the system intentions. As a simple example of using a nested user model, consider a system which always says "How may I help you?" at the start of a dialogue. It need worry little about accurate generation of the utterance after a few runs with the same user since the system will eventually believe that the user believes that the system always intends to start with "How may I help you"

Since dialogue happens in alternating turns, the system constructs a time-tree by adding a choice node for its own turn, then adding a chance node for the uncertain mental state of the user, if that state has a determining effect on the user's choice. Then a further choice node is added for the user's expected choice. This continues recursively, using a further level of the nested belief model at each turn, until all of the outcomes of the dialogue have been generated. Choice nodes are produced by expanding the dialogue plan, which is hierarchically structured, similar to NOAH plans [4]. To represent different utterance generation strategies, alternative decompositions can be given for each dialogue act strategy. For instance, an ellipsis might be used in eliciting a flight date by asking "on which date?", or a fuller utterance "on which date would you like to depart". Two rules are used:

```
decomp(date-ask,[date-ask-elliptical]).
decomp(date-ask,[date-ask-full]).
```

To represent the different strategies in confirmation dialogues, alternative decompositions are used for information eliciting subdialogues. For asking a flight date, the two following decompositions are used:

```
decomp(date-dialogue,[date-ask,date-inform,date-confirm]).
decomp(date-dialogue,[date-ask,date-inform]

decomp(date-confirm,[declare-date,confirm]).
decomp(date-confirm,[declare-date,deny]).
```

Once the time-tree has been constructed, a utility calculation is used to decide between strategies at the current choice node. The utility is calculated from the value of the completed task, and from the cost of each of the strategy choices used in the tree path, in terms of the amount of time taken for each strategy. These costs are specified as part of the domain rules. Using the rule of "maximize expected utility", the expected utility for a time-tree is calculated by recursively processing the choice and chance nodes. For the choice nodes, the utility is the utility of the branch chosen at that node, which each agent attempts to maximise. For the chance nodes, the utility is the expected utility over the probability distribution, which is the sum of the utility of the branches, each weighted by their probability. Finally, to decide a strategy for the current turn, the system chooses the branch at the first choice node that has the maximum expected utility according to this calculation.

3. Probabilistic Plan Recognition

An important step in constructing the time-tree is the plan recognition step, whose purpose is to fit the system or user utterance into a plan structure based on the dialogue context. Previously, plan recognition algorithms for dialogue have not been treated probabilistically [5], but the relative likelihood of the plan hypotheses is critical in making dialogue strategy choices. In the general plan recognition problem, probabilistic reasoning uses a Bayesian network [6]. However, in dialogue, for ease of recognition, speakers usually maintain the focus of a task, meaning that nodes are added to the plan in a bottom-up left-to-right order. As a result, plan recognition is much simpler as nodes are added to the first available position if the plan tree is not full. This requires no plan recognition. If however, the plan structure is full, it is expanded upwards from the root to produce one of several recognition hypotheses, according to the choice of parent for the root node. Implementation of this approach is very simple, since all that is required is a list of alternative parents and their probabilities for any particular dialogue act or internal task node. For lower level recognition, from the speech signal to the dialogue act level, a similar, "n-best" style list can be used to build plan hypotheses.

Plan recognition rules are part of the agent's mental state, and so are encoded within the nested belief model. For example, the system's strategy choice must use the plan recognition rules at the second level of nesting to predict the likely interpretations of the hearer.

4. The Law of the Most Probable and Robust Generation

The use of robust strategies relies on a phenomenon that is called here the "law of the most probable". This law states that when the cooperative hearer of an utterance has several hypotheses for the speaker's intention and can act on only one of them, he must choose the most probable of the intentions, since this will maximise his expected utility. This law is very useful in robustly choosing an utterance, since the speaker need only communicate just enough that his intention is placed as the most probable one in the hearer's mind. As an example in utterance generation, consider a problem of definite description generation. A robot agent and a human are cooperating to repair the adjustment nut on a machine. There is a big spanner and a small spanner in the workshop, of which the robot usually uses the small spanner. The robot would like to use a short, but robust enough definite description in asking the human to pass the small spanner. His choices are:

1. "Pass me the small spanner please"
2. "Pass me the spanner please"

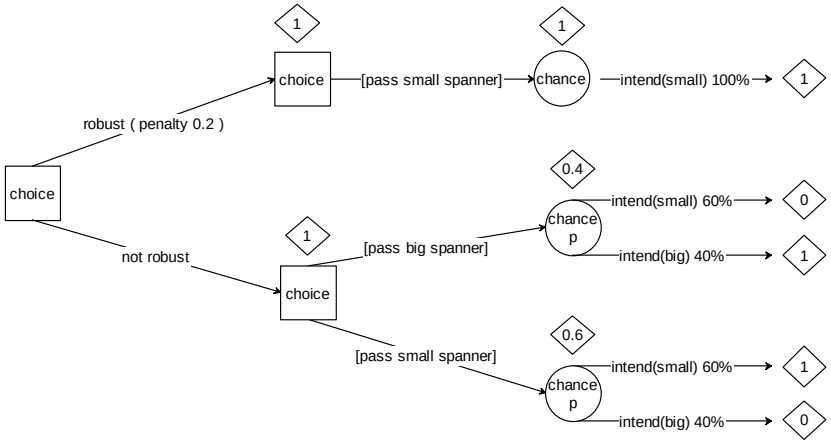


Figure 1: A time-tree for robust generation

Utterance 1 is robust enough to always succeed. Utterance 2 is shorter, but less robust. The planner generates the time tree shown in figure 1 for deciding whether to use the robust form or not. The user model estimates that the user holds a plan recognition rule that estimates the system intention of having a small spanner with a probability of p=0.6. The expected utility value for the robust option is then 0.8 since a reward of 1 is attained at the end of the plan, and since using a robust utterance has a small penalty in time-to-completion. The non-robust option has a utility of 1 since the reward is also attained but at no penalty, and so the agent chooses the non-robust option in this case.

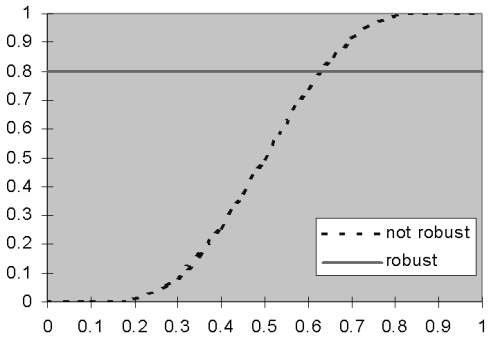


Figure 2: Variation in utility of robust and non-robust strategies with probability values in the user model

The interesting property of this problem is the sensitivity of the agent's decision to the value of p in the user model. If p is anywhere greater than 0.5 the agent should choose the non-robust form. If p is anywhere less than 0.5 the agent should choose the robust form. Having an accurate user model is then necessary to make an accurate strategy decision, and the plan recognition rules and recognition mechanism must be probabilistic.

Figure 2 shows the utility estimated by the system for the two strategies by including estimation error. With no assumed error in the user model, the non-robust utility curve is a sharp step function. However, since the user model is often based on sampling a finite set of runs of the system with the same user, which determines the estimate of p, an error is introduced by treating each run as a Bernoulli trial. For an example of ten runs, a smoother "s" curve is obtained for the non-robust strategy, with the result that the system decides to use the strategy only when p is safely above 0.6, rather than at 0.5. As the user model develops, the curve sharpens, and the system can more safely choose less robust strategies. Therefore, the plan recognition system records error values as well as probability estimates in the plan recognition rules.

5. Further Examples

Converse to the generation problem, in which the system must clarify its own intention, another application of the planner is in clearly eliciting the user's intention. This can be achieved by asking for repetition, rephrasing, or by confirmation. In this example, a train passenger, at Heuston Station in Dublin, asks:

A: Could I have a ticket to Cork please?

The intention is underspecified since it doesn't mention the starting station (there are two possible stations in Dublin), but since the passenger is at the Heuston ticket desk, it is most likely that he wants to travel from Heuston. There is a large penalty for recognition error, since the passenger will be given the wrong ticket, but there is also quite a small probability of this error since the passenger most probably means Heuston. Therefore, the problem is particularly sensitive to the accuracy of the user model. A time tree can be constructed for this problem, and once again there is a user model variable that determines the strategy choice, namely the plan recognition rule that infers the two alternatives from his utterance.

The planner has also been applied to the problem of mixed initiative dialogue, where the system must decide whether or not to act upon a possible user intention. For example, it might be wasteful to always ask train passengers if they want a window seat, if that is rarely their intention. On the other hand, if a passenger does not ask because he thinks they are unlikely to be available, an opportunity is missed. A time-tree (figure 3) can be constructed to solve this problem for the system's decision, using user model values for the probability that a passenger wants a window seat, and the nested value in the user model that represents the passenger's expectation of whether a window seat is available.

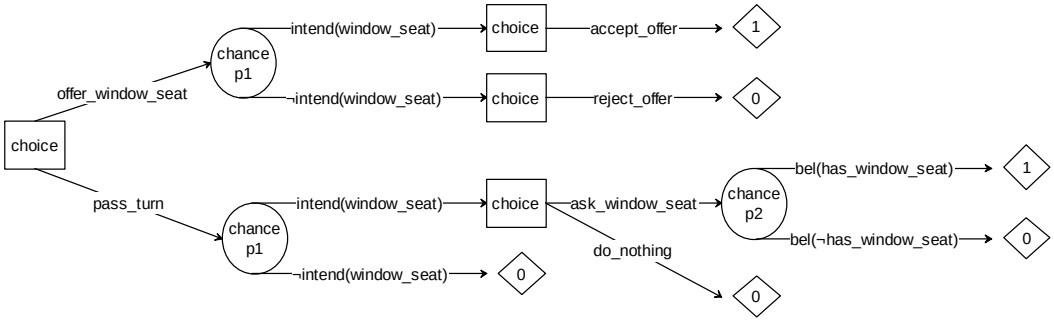


Figure 3: A time tree for robust mixed-initiative planning

6. Discussion

For planning with small numbers of states such as for a typical train reservation dialogue system controlled by a state-based dialogue manager, online planning is not neccessary, since functions can be derived from hand-drawn time trees at design time for each expected state, with variables to taken from the user model. As such the time trees can be used as an analytical tool by the system designer. For example, in the generation dialogue example the functions is:

```
use_robust(UserModel um) {

if (um.p > 0.5)
        return true;
else
        return false;
}
```

For a more complex time-tree, such as figure 3, a nested if statement is used to deal with two user model variables. For finite state problems, reinforcement learning (eg [7]) is a similar approach to deciding strategies at design time, but since it uses a static policy, it is inapplicable to problems that use a dynamic user model.

For more complex planning problems, there may be deeper plans, more complex sets of intention hypotheses, and more complex sets of choices for the system at each turn, the combinations of which are better represented with plan rules rather than fixed sets of states. For example, in repairing a machine, an agent may ask for some superglue because he has developed a novel plan to hold the adjustment screw in place. Recognising that intention has to do with the complex configuration of the machine at hand, mixed with the dialogue acts that help to choose between the hypotheses, and cannot be recorded as a stored state at design time.

The following usage scenarios are envisioned for the planner:

Fully static system The time-trees can be used at design time to hardcode strategy choices for fixed-state systems, with a static user model gathered from user surveys or wizard-of-oz experiments. For example a study of how often passengers desire a window seat determines the initiative policy for asking whether they require a seat. Such analysis would be particularly valuable for time-pressured environments such as in emergency diagnosis and response, and can be applied equally to well to determine the dialogue policy of a human operator.

Finite state with dynamic user model Using functions derived from the time-tree at design time, a dynamic user model, tailored for repeated use by the same user can be instantiated in the field and used with the stored functions.

Fully dynamic system Dynamic problem solving agents produce novel plans with mixed physical and dialogue acts, for which efficient dialogue plans need to be generated on the fly using the planner. The user model adapts dynamically. A user-adaptive tutorial dialogue system, an interactive tour guide, or a computer application troubleshooting assistant must produce efficient yet robust responses. As an example, an interactive tour guide could efficiently adapt to a user who likes both Italian restaurants and record shopping by generating tour plans that could not be effectively retrieved from memory.

7. Conclusions

A dialogue planner was described that can decide whether a robust but costly strategy should be used. It has been implemented, and three example problems have been explored – generation of robust utterances, use of robust strategies in identifying the user intention, and deciding when to take the initiative in opening subdialogues.

8. References

[1] Walker, M., Litman, D., Kamm, C., Abella, A., Evaluating Spoken Dialogue Agents with PARADISE: Two Case Studies .in *Computer Speech and Language*, 12(3), 1998.

[2] Walsh, M., Kelly, R., O'Hare, G.M.P., Carson-Berndsen J., Abu-Amer,T., A Multi-agent Computational Linguistic Approach to Speech Recognition. in proceedings *18th International Joint Conference on Artificial Intelligence*. 1477-, 2003.

[3] Rao, A., Georgeff, M. Modelling Rational Agents Within a BDI Aarchitecture. in proceedings *Second Conference on Knowledge Representation and Reasoning* 473-484, 1991.

[4] Sacerdoti, E., *A Structure for Plans and Behaviour*. Elsevier, New York, 1977.

[5] Carberry, S., *Plan Recognition in Natural Language Dialogue*, MIT Press, 1990.

[6] Charniak, E., Goldman, R., A Bayesian Model of Plan Recognition. *Artificial Intelligence* 64 (1) 53-79, 1993.

[7] Roy, N., Pineau, J., Thrun, S., Spoken Dialogue Management using Probabilistic Reasoning. in proceedings *38th Annual Meeting of the Association for Computational Linguistics*. 2000.