

# **4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems**

**Edinburgh, Scotland  
August 1, 2005**

**Proceedings edited by Ingrid Zukerman,  
Jan Alexandersson and Arne Jönsson**



## 4th IJCAI Workshop on KNOWLEDGE AND REASONING IN PRACTICAL DIALOGUE SYSTEMS

This is the fourth workshop on Knowledge and Reasoning in Practical Dialogue Systems. The first workshop was organised at IJCAI-99 in Stockholm,<sup>1</sup> the second workshop took place at IJCAI-2001 in Seattle,<sup>2</sup> and the third workshop was held at IJCAI-2003 in Acapulco.<sup>3</sup>

The current workshop includes research in three main areas: dialogue management, adaptive discourse planning, and automatic learning of dialogue policies. Probabilistic and machine learning techniques have significant representation, and the main applications are in robotics and information-providing systems.

These workshop notes contain 12 papers that address these issues from various view-points. The papers provide stimulating ideas and we believe that they function as a fruitful basis for discussions and further research.

The program committee consisted of the colleagues listed below, who were assisted by three additional reviewers. Without the time spent reviewing the submissions and the thoughtful comments provided by these colleagues, the decision process would have been much more difficult. We would like to express our warmest thanks to them all.

---

<sup>1</sup>Extended versions of selected contributions were published in a special issue of ETAI, the Electronic Transaction of Artificial Intelligence <http://www.ida.liu.se/ext/etai/>

<sup>2</sup><http://www.ida.liu.se/~nlplab/ijcai-ws-01/ijcai-ws/>

<sup>3</sup>see <http://www.ida.liu.se/~nlplab/ijcai-ws-03/ijcai-ws/>

## **Program Committee**

Johan Boye, Telia Research, Sweden  
Sandra Carberry, University of Delaware, USA  
Peter Heeman, Oregon Graduate Institute, USA  
Eric Horvitz, Microsoft Research, USA  
Kazunori Komatani, Kyoto University, Japan  
Staffan Larsson, Göteborgs Universitet, Sweden  
Diane Litman, University of Pittsburgh, USA  
Michael McTear, University of Ulster, UK  
Norbert Reithinger, DFKI, Germany  
Candy Sidner, MERL, USA  
David Traum, USC Institute for Creative Technology, USA

## **Additional Reviewers**

Thomas Kleinbauer, DFKI, Germany  
Tim Paek, Microsoft Research, USA  
Joel Tetreault, University of Pittsburgh, USA

## **Organizing Committee**

Ingrid Zukerman (Chair)  
School of Computer Science and Software Engineering  
Monash University  
Clayton, Victoria 3800, Australia  
email: [ingrid@csse.monash.edu.au](mailto:ingrid@csse.monash.edu.au)

Jan Alexandersson (Co-chair)  
German Research Center for Artificial Intelligence, DFKI GmbH  
Stuhlsatzengasseweg 3  
D-66 123 Saarbrücken, Germany  
email: [janal@dfki.de](mailto:janal@dfki.de)

Arne Jönsson (Co-chair)  
Department of Computer and Information Science  
Linköping University  
S-581 83 Linköping, Sweden  
email: [arnjo@ida.liu.se](mailto:arnjo@ida.liu.se)

## Table of Contents

### K. Eliasson:

- Towards a Robotic Dialogue System with Learning and Planning Capabilities* ..... 1

### M. Niemann, S. George and I. Zukerman:

- Towards a Probabilistic, Multi-layered Spoken Language Interpretation System* ..... 8

### J. Wyatt:

- Planning to Resolve Ambiguous References to Objects* ..... 16

### M.E. Foster and M. White:

- Assessing the Impact of Adaptive Generation in the COMIC Multimodal Dialogue System* ..... 24

### P. Wärnestål:

- User Evaluation of a Conversational Recommender System* ..... 32

### K. Komatani, R. Hamabe, T. Ogata and H.G. Okuno:

- Generating Confirmation to Distinguish Phonologically Confusing Word Pairs in Spoken Dialogue Systems* ..... 40

### S. Lesch, T. Kleinbauer and J. Alexandersson:

- Towards a Decent Recognition Rate for the Automatic Classification of a Multidimensional Dialogue Act Tagset* ..... 46

### B. McEleney and G. O Hare:

- Efficient Dialogue Using a Probabilistic Nested User Model* ..... 54

### B. Inouye and A. Biermann:

- An Algorithm that Continuously Seeks Minimum Length Dialogs* ..... 62

### J. Henderson, O. Lemon and K. Georgila:

- Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data* ..... 68

### J. D. Williams, P. Poupart and S. Young:

- Factored Partially Observable Markov Decision Processes for Dialogue Management* ..... 76

### M. Frampton and O. Lemon:

- Reinforcement Learning of Dialogue Strategies using the User's Last Dialogue Act* ..... 83

# Towards a Robotic Dialogue System with Learning and Planning Capabilities

Karolina Eliasson

Linkoping University

Department of Computer and Information Science

SE - 581 83 Linköping

karel@ida.liu.se

## Abstract

We present a robotic dialogue system built on case-based reasoning. The system is capable of solving references and manage sub-dialogues in a dialogue with an operator in natural language. The approach to handle dialogue acts and physical acts in a unison manner together with the use of plans and sub-plans makes the system very flexible. This flexibility is used for learning purposes where the operator teaches the system a new word and the new knowledge can directly be integrated and used in the old plans. The learning from explanation capability makes the system adaptable to the operator's use of language and the domain it is currently operating in. The implementation of a case-based planner suggested in the paper will further increase the learning and adaptation degree.

## 1 Introduction

Human-Robot Interaction is a broad and interesting area which deals with the interaction between a human and a physical embodied robot. In our work we have focused on interaction in natural language with an autonomous Unmanned Aerial Vehicle (UAV). In a dialogue with such a robot, the dialogue manager must be able both to interpret the utterance from the operator, receive additional information to be able to react correctly to the utterance, and perform a sequence of actions. To do this, it has to distinguish between *dialogue acts* and *physical acts*. A dialogue act can be to ask a clarifying question to the operator and a physical act can be to perform the high-level command `land`. These acts must be executed in a correct order to solve the problem at hand. If the information is not sufficient for the dialogue manager to complete the task it has to ask for complementary information from the operator. In such a case a discourse model is needed to solve anaphoric references and to manage sub-dialogue.

In case the robot will be used by the same operator, it can be convenient if the robot can adapt to the operator's use of language. In that case a flexible control system is needed which gives the operator the opportunity to explain what a word means and how a task, new to the dialogue system, shall be performed. In this manner, the dialogue system can be adapted not only to the current operator's use of language

but also to the different tasks that can be of interest in different flight scenarios. To further increase the usefulness of the dialogue system, we would like it to be able to perform mixed-initiative planning with user interaction and reuse old plans and experiences.

In this paper we will describe a dialogue system which uses case-based reasoning (CBR) to find suitable responses to the utterances from the operator and messages from the robot. CBR is a form of machine learning where the system stores problems and their corresponding solutions in a case base. When a new target case enters the system, it searches the case base for similar cases. When the most similar case is found, its corresponding solution is adapted to the new target case and the new solution is returned. The new target case and its solution is then stored in the case base for future use. See for example [Aamodt, 1994] for an overview. We are also addressing case-based planning (CBP) where a generative planner is used when there are no suitable plans in the case base. The planner should not plan a new solution from scratch but reuse the old plans as much as possible.

CBR provides our dialogue system with a simple and modular design where both the dialogue acts and the commands regarding the physical acts can be represented in a uniform way. New functionality is directly added by writing new cases and storing them in the case base. New domain knowledge similar to existing knowledge can be added to the system in a simple manner. It can directly be used by the system without any additional changes to the case base or the case-base manager, due to the flexible and adaptable nature of the CBR design. This provides us with the facility of letting the system incorporate new information, such as new words or knowledge about the physical world, into the system. This knowledge can then directly be used by the cases in the case base, hence giving the system mechanisms for updating its own knowledge and increasing its performance. The new information can be obtained from dialogue with an operator. Because phrase matching is necessary both in CBR and in discourse modeling, in the latter to allocate incoming new phrases to the correct dialogue thread, it makes CBR and discourse modeling a suitable combination without producing any additional overhead.

Our dialogue system CEDERIC, Case-base Enabled Dialogue Extension for Robotic Interaction Control, which addresses the above mentioned features is still under construc-

tion and this paper will discuss both implemented features and some ideas and work in progress. Most of the parts, such as the CBR framework which uses existing plans, the discourse model, and the learning from explanation dialogues are implemented and tested in a simulator, but the work with the planner is still ongoing.

## 2 CEDERIC

CEDERIC is a dialogue system designed for dialogue with a physical robot, in particular the WITAS autonomous UAV. The WITAS project focuses on the development of an airborne computer system that is able to make rational decisions about the continued operation of the aircraft, based on various sources of knowledge including pre-stored geographical knowledge, knowledge obtained from vision sensors, and knowledge communicated to it by data link [Doherty *et al.*, 2000]. The UAV used in the project is a Yamaha RMAX helicopter which an operator can control by high level voice commands or by written commands. The operator can ask the UAV to perform different tasks and request information from it.

CEDERIC consists of a *parser*, a *case base*, *domain knowledge*, a *discourse module* and a *case-base manager* as shown in Figure 1. The parser parses the sentence given by the op-

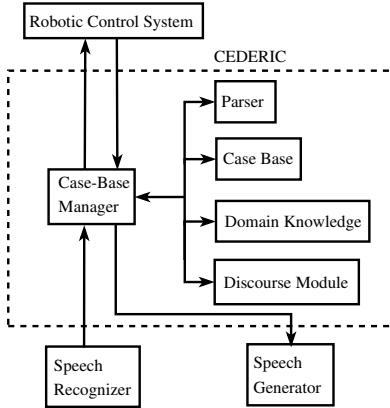


Figure 1: Architecture of CEDERIC.

erator and generates a parse tree. The parse tree is matched against cases in the case base by the case-base manager. The discourse module is responsible for maintaining a discourse model of the dialogue so far to be able to interpret the operator's sentences in the right context. The discourse model helps the system to interpret references which may refer to sentences earlier in the dialogue. The domain knowledge contains an ontology of the world as the robot knows it and a categorization of the world items. The purpose is twofold. It serves as a world representation which gives CEDERIC knowledge about which buildings there are in the known world, what kind of buildings they are, where they are places, and their attributes such as color and material. It also gives CEDERIC fundamental knowledge about categorization e.g. which items that can be called buildings in the dialogue and which can not.

```
(PHRASE1
(VERBPHRASE_IMP }
(VERBPHRASE_IMP7 (FLY FLY) (TO TO)
(NP_FLY_TO
(NP_BUILD (DET THE)
(NOUN_BUILD
(NO_CAT SUPERMARKET))))}))}
```

Figure 2: Parse tree with an unknown word.

The operator can choose to use either speech or text for the input to the dialogue system. The speech recognizer used is the off-the-shelf product Nuance and the speech generator used is one of the off-the-shelf products Festival or Bright-speech. When learning a new word using speech recognition, one can choose between having a considerably bigger grammar for the speech recognition than the dialogue manager and only consider learning in the dialogue manager, or provide the new word in text form in the learning phase and then compile it into the speech recognition grammar if that can be done at runtime. We have chosen the second approach where the unknown words are provided in text and the learning phase extends the grammar and lexical rules in the parser within CEDERIC. The grammar and the lexical rules are the same as those for the speech recognizer, so the extensions should be easy to integrate with the speech recognition system if it allows runtime updates of the grammars.

Regardless of whether a speech recognizer is used or not, a sentence from the operator arrives to CEDERIC in plain text format. It is parsed, processed in the CBR engine and returns either a new phrase in text format to be sent to the speech generator, or a request to the robotic control system. The robot requests are in a format similar to KQML [Finin *et al.*, 1994]. The response the robot sends back to CEDERIC is also in a KQML-like format. Those messages are sent to CEDERIC on a different channel than the one the operator uses. This is because the response should not be parsed in any way, but be processed directly in the CBR engine.

### 2.1 Parser

The text string that results from the speech recognizer or is obtained directly from the operator if he or she provides the input in text form, is parsed by a chart parser. The chart parser returns a parse tree of the input sentence. If some of the words are unknown to the parser it provides all possible solutions where the unknown words are first labeled with the possible label and then with the label *no\_cat*. Figure 2 shows an example of a parse tree where the word SUPERMARKET is unknown. It has been categorized with the possible category *NOUN\_BUILD*. When there is an unknown word in the sentence, all the possible parses are collected in a parse tree which is sent to the case-base manager. The case-base manager can then try the different possible matches and see if it finds a suitable match with one of its cases.

### 2.2 Case Design

To be able to reuse parts of a solution, each solution to a sentence is divided into smaller units, each of which performs an isolated part of the solution. These units are called *atomic*

*cases*. For the robot to know how to construct the whole solution out of the atomic cases it needs a plan. Already known and tested plans are stored in *plan cases*. Both plan cases and atomic cases have a *problem part*. This part of the case consists of the name of the case and the problem information that the target case need to have to match the case. The problem information can be either:

- a parse tree
- the result from a previously executed atomic case or plan case
- a response from the robotic control system in KQML-like format.

They also have a *discourse update part* which updates the discourse model of the dialogue with new information about the current dialogue.

The *solution part* of a plan case is a plan. The plan consists of the name of other plan cases and atomic cases. The solution part of an atomic case consists of procedures which return an answer. An atomic case can for example be *get-referenceid* which takes a parse-tree of a sentence like *white church*, checks its domain knowledge for a match and returns every matching reference id to such a building. The next item in the plan could then use this information to produce another step in the plan.

### 2.3 Case Matching

When a case match is performed, the problem information of the case is matched against the target case. The whole target case does not have to match the problem information completely to be considered a perfect match. It is sufficient that all the problem information occurs in the correct order in the target case. This approach makes the cases more general and does not leave it to the case-base manager to adapt each target case in a suitable way to obtain the new target case match. This is very useful when you want a case to be able to match several similar target cases. An example in our domain is the command *fly to the red hospital*. The parse tree for this sentence looks like this:

```
(PHRASE1
  (VERBPHRASE_IMP
    (VERBPHRASE_IMP7 (FLY FLY) (TO TO)
      (NP_FLY_TO
        (NP_BUILD (DET THE)
          (ADJPHRASE_BUILD (ADJ_COLOR RED))
            (NOUN_BUILD HOSPITAL))))))
```

If the whole parse tree where to be matched, we would need a new case for every possible combination of color and building. Moreover, we would have to write new cases for the similar cases where another attribute instead of color is given, or where no attribute at all is provided in the sentence. This could however be solved with adaptation in the case-base manager before the match but the more similar target cases there are, the more information about adaptation of the target cases has to be hard coded into the case-base manager with no possibility to learn or expand the knowledge by experience. We have taken the decision to let the problem information leave out some of the specific information in the match and

act as a general case. The problem information for a case matching the example above can look like this:

```
(PHRASE1
  (VERBPHRASE_IMP
    (VERBPHRASE_IMP7 (FLY FLY) (TO TO)
      (NP_FLY_TO ))))
```

The cases are ranked dependent on how well they match the target case. The ranking includes two parameters:

- How well the target case covers the case.
- How well the case covers the target case.

Both parameters are expressed on a scale from 0 to 100. The first parameter is obviously the most important one and the cases are firstly sorted by it and secondly by the the second one. That is, we want cases for which the information in the target case are enough to fulfill the requirement of the case and if we find several such cases, we prefer cases which match as much of the target case as possible.

### 2.4 Discourse Module

For a dialogue in natural language to run smoothly, the participants have to know the history of it. If a computer dialogue system will be able to work properly in such a natural dialogue with a human user it has to maintain a discourse model of the dialogue so far to be able to interpret the utterances of the user in the right context. The discourse model helps the system to interpret references to utterances earlier in the dialogue.

The discourse model implemented in CEDERIC is a slightly modified version of the one described in [Eliasson, 2005]. Its design is highly inspired by the discourse model presented in [Pfleger *et al.*, 2003] for the SmartKom project. It is built up of four different objects, that are linked to one another in a hierarchical manner to represent the meaning of the dialogue.

*The linguistic objects.* These objects are furthest down in the chain of objects and thus most specific on the word level. They contain information of how the nouns in the dialogue where uttered. They could for example have been references by the word *it* or by a noun and a determinant.

*The discourse objects.* These objects contain the different nouns together with their attributes mentioned in the dialogue. A discourse object can also be composite. An enumeration of several objects can be seen as a discourse object representing the enumeration as such, and this object contains the enumerated objects as its children. This gives CEDERIC the opportunity to understand references referring to the order of the enumerations, e.g. the first one. The discourse objects have a link to the corresponding linguistic object.

*The dialogue objects.* These objects group those sentences having the same direct goal, with their respective associated information. The sentence *fly to the hospital* gives for example, when it is executed, a dialogue object grouping the sentences *fly to the hospital*, *ok* and *I am at the*

hospital now. If any sub-dialogues come up, they will be saved in a new dialogue object with their direct goal to clarify some matter in the dialogue. Dialogue objects contain information about the topic of the dialogue, which discourse objects were created due to the utterances, and which future utterances this dialogue object expects before considering the dialogue or the sub-dialogue to be completed. These expectations on future dialogue are saved in a modified *initiative-response (IR) unit* [Ahrenberg *et al.*, 1991]. Unlike the original IR-units described by Ahrenberg, IR-units in our context can contain more than two subelements. This is necessary because they shall also be able to represent the response from the robot when the system sends a request. The *fly to the hospital* example above shows such an example.

*The global focus space.* The various objects in the dialogue layer which belongs to the same dialogue, including sub-dialogues, are grouped together in a top object called the global focus space. It contains information about the main topic of the dialogue and about which dialogue objects belong to it. Each global focus space also keeps track of the discourse object last mentioned, to be able to resolve references such as *it*. This is known as the *local focus stack*. The last mentioned discourse object is said to be in focus.

To keep track of the current dialogue in focus, CEDERIC saves the different global focus spaces in a stack called the *global focus stack*. The global focus space on top of the stack is said to be the one in focus. If every IR-unit belonging to a global focus space is closed, that is, has received all its subelements, the global focus space is marked as closed and removed from the stack. Several dialogues can be open and ongoing at the same time and are thus members of the stack but only one dialogue can be in focus at the same time.

When a new sentence is recognized and matched with a case in the case base, the discourse update part of the case is executed. This information creates new objects in the different layers and links them together to reflect the identified dialogue.

## 2.5 Case-Base Manager

The case-base manager is the main engine responsible for the data flow in the dialogue system. It matches the target case with the cases in the case base as described in section 2.3, selects the first case in the ranked case list, and evaluates the first item in the selected case's plan. During the execution of the plan it stores the plan history. If one of the plan items in the current plan does not match the input from a previously executed plan item, the case-base manager selects the next case in the ranked case list and checks if the history equals the first part of that case's plan. If so, the case-base manager can switch to that plan and continue with it. In this way, CEDERIC can cope with information not known from start. This is important when the dialogue system has a sub-dialogue with the operator to clarify some information. With the operator in the loop, the data gets unpredictable, because CEDERIC can not know which answer the operator will provide in advance

and hence the plan chosen from the beginning can be found to be wrong.

The case-base manager is also responsible for keeping track of the different ongoing dialogues. When a new sentence arrives from the operator, it could be one of the following cases:

- The operator continues the current dialogue, possibly by the start of a sub-dialogue.
- The operator returns to an older non-completed dialogue with is not presently in focus.
- The operator starts a new dialogue, possibly without ending the recent dialogue properly.

It is important to recognize which of these three cases it is, to be able to provide the correct discourse for the evaluation of the sentence. The case-base manager starts by checking if there are any ongoing plans and in that case the case-base manager tries to go on with it, possibly by changing to another plan as described above. If it does not give any result, the manager tests to execute one of the saved older non-completed plans with its corresponding discourse. If that fails as well it starts a new dialogue with an empty discourse model.

## 2.6 Learning from Explanation

CEDERIC can ask the operator for guidance if the parsed sentence contains the category *no\_cat*. As described in section 2.1, the parser generates all possible parses with both the *no\_cat* category and the possible category. To be able to catch those *no\_cat* parses, CEDERIC needs to have cases that match both the sentence where the *no\_cat* was found and the *no\_cat* itself. By providing CEDERIC with such cases, it can ask the operator questions that makes him or her explain how CEDERIC should react. We take the parse tree in Figure 2 as an example. It has been generated from the sentence *Fly to the supermarket* where the word *supermarket* is not in the parser's vocabulary. The case base contains a case where the problem part is the parse tree in Figure 2, but with the word *supermarket* omitted (obviously, because we want the case to match all unknown words!). This case is fully covered by the target parse tree which characterizes a good match and its plan is executed, which, depending on the answers from the operator, can result in the dialogue in Figure 3. After the execution of this dialogue, the vocabulary of the parser is extended to also contain the word *supermarket* which is a building. The newly gained information such as the color and the category of the supermarket will be saved in the domain knowledge as well. The next time the operator wants the robot to fly to the supermarket, the sentence will be correctly parsed and the case that matches such a sentence will provide a correct solution.

The dialogue shown above where the operator guides the UAV to an unknown building can be used with any word that the operator wants to use as an identifier for a particular building. This particular plan case is however not general enough to handle other types of words, which are not buildings. They can be handled analogously by using other plan cases which describes the sub-dialogue suitable to learn CEDERIC how to react properly on them.

---

Operator: Fly to the supermarket.  
 CEDERIC: I do not know what a supermarket is.  
     Is it a building?  
 Operator: Yes.  
 CEDERIC: Can you give me a reference object near the supermarket so I can start by flying to it?  
 Operator: It is near the red hospital.  
 CEDERIC: Ok, I will start by flying to the hospital and then I will ask for more guidance.  
 CEDERIC: I am at the hospital now.  
 CEDERIC: In which direction shall I fly to look for the supermarket and what characteristics does it have?  
 Operator: Fly eastwards and look for a blue building.  
 CEDERIC: I start fly and look for the supermarket.  
     I will stop when I see it.  
 CEDERIC: I have found the supermarket now and hover over it.

---

Figure 3: An example of a teaching situation between the operator and CEDERIC.

The main approach is generally applicable and a similar solution has been tested for teaching CEDERIC new verbs, which are synonyms for already known commands. The sentence `Go to the red hospital`, where the word `go` is unknown to the parser and hence categorized with the category `no_cat` and the potential category `fly`, is matched with a case whose response is to ask the operator if the words `fly` and `go` are synonyms. If the operator answers yes, CEDERIC will process the command as a `fly` command and put the new word `go` in the vocabulary. It also creates a new case which changes the word `go` to `fly` and calls the plan case for `fly` commands.

It should also be possible to learn CEDERIC new composite commands in a similar fashion, if the building blocks are already known. This is however not tested yet.

## 2.7 Planner

Case-based Planning (CBP) is a CBR field where the solution to a problem is stored as a plan. When a new problem enters the system, the case base is searched for a case whose plan can easily be adapted to the new problem. The found plan may partly be used while the remaining plan items are taken from another case or by using a generative planner. A survey of CBP and CBP systems can be found in e.g. [Munoz-Avila *et al.*, 1998] and [Spalazzi, 2001]. This planning technique can be very useful in CEDERIC for solving new, unseen composite problems. CEDERIC does already handle plans and it can switch between plans if the current one turns out to be wrong, but with the use of CBP the system can also combine several different plans into a composite one. It can also solve problems similar to those which have a solution by using the known solution plans and exchange the non suiting plan items to other plan items suggested by the generative planner. An example of the use of CBP is when CEDERIC knows how to solve the problem `Fly to the hospital` where the solution is to first look up the position of the hospital in the

domain knowledge and then send a request to the robotic control system to fly to that position. We assume it also knows how to ask the robotic control system for a position of a car. If the operator gives the command `Fly to the parked car`, CEDERIC can create a plan which first asks the robotic control system for a position of the car and then use this position as input to the second part of the plan for flying to the hospital.

CBP can also be useful for CEDERIC to understand implicit information and to extend a plan when appropriate. One example is if the operator tells CEDERIC to fly somewhere but the UAV has not taken off yet. Then CEDERIC can understand the implicit command to take off and then perform the `fly` command. To make the planning more reliable, the dialogue system can report the plan to the operator and ask for confirmation before executing it.

Together with dialogue features already provided by CEDERIC, CBP can be used not only to perform planning by itself but also to implement mixed-initiative planning with user interaction. The system can ask the operator questions and the planning can be performed in cooperation with the operator. This will increase the usefulness of the system and the system will learn from it and can reuse the plans in the future. CBP can in particular be useful for developing the learning from explanation features described in section 2.6. further, where the system and the operator can plan a new learning sequence.

It has been argued that the CBP technique, where already stored plans are adapted to a new problem, does not add anything considering time complexity compared to generating new plans from scratch [Nebel and Koehler, 1995]. In our case however, low time complexity is not the most important reason for reusing old plans. When dealing with real world problems, several factors can affect the result and all factors may not be included in the problem formulation. Therefore, it is safer to use an already proven plan to a large extent than to generate a new one.

## 3 Implementation and Testing

The case base manager and the parser are implemented in lisp. The case base, domain knowledge and discourse model are implemented in The Knowledge Machine [Clark and Porter, 2004]. The planning with a generative planner and the use of several case plans to achieve a plan to a new target problem is still work in progress.

CEDERIC has been tested connected to a simulated UAV situated in a simulated environment. The simulated UAV can perform high level actions such as flying to a building with a certain unique identity, flying in a certain direction, take off, land, ascend and descend. It reports the results of an action and also reports if it observes any buildings as it flies. It is able to answer status questions such as the current altitude, velocity and heading, etc [Andersson, 2005]. The interface between CEDERIC and the simulator is very similar to the interface offered by the actual WITAS UAV, in order to make it straightforward to use CEDERIC during actual flights. The simulation of UAV movements and environment only goes to the level of detail that is required for this interface, and does not go down to the level of exact flight dynamics.

---

Operator: Fly to the school.  
 CEDERIC: I have two schools to choose between.  
     Which one do you mean?  
 Operator: Which can I choose between?  
 CEDERIC: You can choose between the one on  
     Harborroad and the one on Mainstreet.  
 Operator: The one on Harborroad.  
 CEDERIC: Ok.  
 CEDERIC: I am at the school now.

---

Figure 4: A dialogue example.

Due to the work in progress status of CEDERIC, it has not yet been rigorously tested in a formal manner. However, it has been tested with various types of dialogue, from simple robotic control commands such as `take off` and `land` to complex dialogue with sub-dialogues and learning from explanation. Figure 3 gives an example of an implemented and tested learning from explanation dialogue. In the tests, the first sentence has been provided in text because the speech recognizer does not yet recognize the new word, but the rest of the dialogue has been tested using a speech recognizer. The answers are all produced by a speech generator. Figure 4 gives an example of a dialogue with several sub-dialogues. It has been tested using speech in and speech out.

The sub-dialogue for finding a unique object is not tightly connected to a specific dialogue but can be used as a sub-plan in every dialogue where the system needs a unique reference identification to the object the operator refers to. Other sub-dialogues can easily be implemented in a similar fashion and used whenever it may be useful.

## 4 Related Work

Conversational CBR (CCBR) is an area within CBR where the user usually wishes to query a database with items, e.g. searching for a computer to buy on an e-commerce site. Aha et. al. provide an overview of CCBR in [Aha et al., 2001]. The e-commerce system ExpertClerk is described in [Shimazu, 2002]. Case Advisor is a generic CCBR system which allows an organization to efficiently author and retrieve solutions from a knowledge database to solve customer problem. It is a commercial tool but some of its features is described in e.g. [Racine and Yang, 1997]. CCBR differs from our work in several essential ways. CEDERIC is capable of learning from experience and saves new cases in the case base for further use, which CCBR systems do not. It is also capable of learning from explanation whereas CCBR systems are not. Another big difference is that the sentences in CEDERIC can be saved in cases of their own and do not have to be coupled to a physical action, i.e. the questions are separated from the items in the case base.

HICUP [Aha et al., 2001] is a CCBR system with integrated planning capabilities developed by Aha et al. HICUP is used to plan noncombatant evacuation operations (NEOs). HICUP uses published military doctrines as well as information from previously performed NEOs to guide the search. As in CCBR, the user gets simple questions to narrow the simi-

lar case set. HICUP does not save its own solutions and thus does not learn from experience more than using the information from previously performed NEOs which is manually fed into the system, in contrast to CEDERIC which saves the solutions for further use. HICUP also differs from CEDERIC in the use of language in the system. HICUP does not provide the user with a rich dialogue and does not learn from explanation.

Within the WITAS project, several dialogue systems with various capabilities have been developed. The first WITAS Dialogue System [Lemon et al., 2001] was a system for multi-threaded robot dialogue using spoken I/O. The DOSAR-1 system [Sandewall et al., 2003] was a new implementation using another architecture and a logical base. This system has been extended into the current OPAS system [Sandewall et al., 2005]. Our work takes a rather different approach than their systems due to the use of CBR, the integration of learning capabilities using adaptation of the cases and learning from explanation. Our system also addresses the issue of planning and reuse of plans in dialogue and robotic actions.

Some work has been done concerning robotic dialogue and learning from explanation. Asoh et al. [Asoh et al., 1997] have developed a robot called Jijo-2, which is able to create a map over its surroundings through conversation with a human teacher. The human teacher can give the robot a description of how the surroundings look like and how to solve wayfinding problems and the robot can ask the teacher questions. Theobalt et al. have created a robot similar to Jijo-2, which uses domain knowledge in the learning process [Theobalt et al., 2002]. Carl [Lopes, 2002] is a robot which has some capabilities of learning from explanation. The operator can tell Carl facts that are stored in the memory of the robot. This information can be used later on in the dialogue. None of these robots use CBR for their dialogue and task planning as CEDERIC does and they can not reuse old plans unless told to learn them in advance as in learning from explanation.

## 5 Conclusions and Future Work

We propose a robotic dialogue system with learning capabilities which make it adaptable with respect to the operator and the problem domain. The system, CEDERIC, is built using CBR techniques and includes a *parser*, a *case base*, *domain knowledge*, a *discourse module* and a *case-base manager*. It is capable of having a dialogue including sub-dialogue such as clarifying questions and can handle references. The dialogue acts and the commands regarded the physical acts are treated in a uniform manner and the dialogue acts contributes to the solution of a problem as much as the physical acts do. The acts are saved in the case base and can be reused at any time. The case base also includes plan cases whose solution is a plan consisting of other sub-plans or atomic actions. The operator can adapt the system by serving as a teacher. The systems learns from explanation and when a new word is learnt, it can easily be used in different contexts. This is done by the flexible plan architecture where composite acts are modeled as plans. Several plans can match the initial problem definition and if it turns out that the selected plan does not match the actual course of events, the system

switches to another more suiting plan.

CEDERIC is tested connected to a simulated UAV and it is able to send and receive messages both from the operator and the robotic control system and to perform a dialogue in natural language. More exhaustive tests will be performed when the system reaches a mature status.

CEDERIC is work in progress and an integrated planner module built on CBP techniques is currently developed. The planner module will provide CEDERIC with advanced learning functions and abilities to understand implicit knowledge that the operator does not provide. The planner performs mixed initiative case-based planning with user interaction where the system is able to reuse parts of plans from several cases in the case base to solve a new planning problem. Using the dialogue features, the operator can guide the search and teach the system new information during a planning action.

We expect CEDERIC to be a full fledged dialogue and planning system which can work in cooperation with the operator in a safe and secure way. When it is mature enough, we expect to test the system connected to the physical helicopter and to demonstrate the system in actual flight.

## Acknowledgement

This research work was funded by the Knut and Alice Wallenberg Foundation, Sweden.

## References

- [Eliasson, 2005] Karolina Eliasson. Integrating a discourse model with a learning case-based reasoning system. In *Proceedings of DIALOR-05*, 2005.
- [Finin *et al.*, 1994] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.
- [Lemon *et al.*, 2001] Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. The WITAS multi-modal dialogue system. In *Proceedings of EuroSpeech*, 2001.
- [Lopes, 2002] L. Seabra Lopes. Carl: from situated activity to language level interaction and learning. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, 2002.
- [Munoz-Avila *et al.*, 1998] Hector Munoz-Avila, Ralph Bergmann Manuela Veloso, and Erica Melis. Case-based reasoning applied to planning tasks. *Case-Based Reasoning Technology from Foundations to Applications*, 1998.
- [Nebel and Koehler, 1995] Bernhard Nebel and Jana Koehler. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artif. Intell.*, 76(1-2):427–454, 1995.
- [Pfleger *et al.*, 2003] Norbert Pfleger, Jan Alexandersson, and Tilman Becker. A robust and generic discourse model for multimodal dialogue. In *Workshop Notes of the IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2003.
- [Racine and Yang, 1997] Kristi Racine and Qiang Yang. Maintaining unstructured case base. In *Proceedings of the Second International Conference on CBR*, 1997.
- [Sandewall *et al.*, 2003] Erik Sandewall, Patrick Doherty, Oliver Lemon, and Stanley Peters. Words at the right time: Real-time dialogues with the witas unmanned aerial vehicle. In *Proceedings of the 26th Annual German Conference in AI*, 2003.
- [Sandewall *et al.*, 2005] Erik Sandewall, Hannes Lindblom, and Björn Husberg. Integration of live video in a system for natural language dialog with a robot. In *Proceedings of DIALOR-05*, 2005.
- [Shimazu, 2002] Hideo Shimazu. ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artif. Intell. Rev.*, 18(3-4):223–244, 2002.
- [Spalazzi, 2001] Luca Spalazzi. A survey on case-based planning. *Artificial Intelligence Review*, 16(1):3–36, 2001.
- [Theobalt *et al.*, 2002] Christian Theobalt, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve. Talking to godot: Dialogue with a mobile robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, 2002.

# Towards a Probabilistic, Multi-layered Spoken Language Interpretation System

Michael Niemann and Sarah George and Ingrid Zukerman

School of Computer Science and Software Engineering

Monash University

Clayton, VICTORIA 3800, AUSTRALIA

{niemann,sarahg,ingrid}@csse.monash.edu.au

## Abstract

We present a preliminary report of a probabilistic spoken-language interpretation mechanism that is part of a dialogue system for an office assistant robot. We offer a probabilistic formulation for the generation of candidate interpretations and the selection of the interpretation with the highest posterior probability. This formulation is implemented in a multi-layered interpretation process that integrates spoken and sensory input, and takes into account alternatives derived from a user's utterance and expectations obtained from the context. Our preliminary results are encouraging.

## 1 Introduction

We present a preliminary report of a probabilistic spoken-language interpretation mechanism that is part of a dialogue system for an office assistant robot called DORIS (*Dialogue Oriented Roaming Interactive System*). Eventually, DORIS is expected to exist in the physical world, participate in dialogue, be able to answer simple questions about its environment, cooperate with the user in achieving a mutual understanding of goals and world state, and achieve goals. The hypothetical dialogue in Figure 1 illustrates some of the capabilities envisaged for DORIS.

- Lines 1-3 illustrate DORIS' understanding that it has to record Sarah's message, and formulate a plan to execute the "telling" action (finding Ingrid and playing back Sarah's message).
- Lines 4-5 show pronoun disambiguation during discourse interpretation, and the ability to judge (to a certain extent) the appropriateness of an answer during response generation (if DORIS had last seen Sarah yesterday, then a more appropriate answer may be "I don't know").
- Lines 6-7 show the envisaged ability to understand indirect speech acts during discourse interpretation, and the ability to formulate a dynamic plan – looking for Sarah.
- Line 8 demonstrates DORIS' understanding of "get Sarah" (line 6): telling Sarah that she is being fetched, and informing Sarah of Ingrid's location.
- Lines 9 and 10 illustrate DORIS' prospective ability to (a) process several sentences, (b) disambiguate between objects in the world (electronic notebook versus paper pad), and (c) act on a portion of the discourse that was

- 1 Sarah: Doris, please tell Ingrid I'm ready for the meeting now.
- 2 DORIS: Certainly. [*Finds Ingrid in her office*]
- 3 DORIS: Hello Ingrid. Sarah said *PLAYBACK MESSAGE*.
- 4 Ingrid: Where is she?
- 5 DORIS: I saw her in the kitchen 3 minutes ago.
- 6 Ingrid: Can you please get her?
- 7 DORIS: OK. [*Goes to kitchen. Sarah is not there. Looks for Sarah, finds her in her office*]
- 8 DORIS: Hello Sarah. Ingrid asked me to fetch you. She is in her office.
- 9 Sarah: I'll be there in a minute. I would like you to carry my notebook, so I don't spill coffee on it.
- 10 DORIS: I understood that you want me to carry your notebook, but not the rest of what you said. Is that enough?
- 11 Sarah: That is fine.
- 12 DORIS: OK. [*Follows Sarah back to Ingrid's office, carrying the notebook*]

Figure 1: Hypothetical dialogue with DORIS

understood, while (d) asking a general question about two portions that were not fully processed. This processing failure could be due to a number of reasons, e.g., speech recognition failure, or failure to associate actions (e.g., spill) or objects in the real world (e.g., coffee) with Sarah's discourse.

Clearly, many of these capabilities are quite speculative. However, by specifying such high aims, we encourage the development of a general framework that will support such capabilities. As seen in the rest of this paper, our current prototype is considerably more modest than the envisaged one. To speed up development, at present DORIS is a virtual agent, rather than a physical robot. Also, for this prototype we are concentrating on the interpretation of one type of spoken sentence (imperatives).

The focus of this paper is on the generation of candidate interpretations from individual spoken sentences, and the selection of a preferred interpretation, where an interpretation consists of a frame-based internal representation of the utterance. This frame forms the basis for the robot's action or the next dialogue turn. For example, Figure 2 shows an interpretation for the request "Get the blue mug". If the location of MUG05 is known, DORIS can act on this request. Otherwise, it can either look for the mug or inquire about its location.

PropositionalContent:
{
Action: fetch_deliver
Agent: DORIS (addressee)
Patient: MUG05 (COLOUR blue)
Recipient: MICHAEL (speaker)
}

Figure 2: Interpretation of “Get the blue mug”

The main idea behind our mechanism is that the best interpretation of a user’s utterance is that with the highest posterior probability. This idea has long been dominant in Automatic Speech Recognition [Young, 2002], and has gained popularity for other aspects of language interpretation in the last decade [Miller *et al.*, 1996]. In addition, the use of contextual information to improve speech recognition performance has also become increasingly popular, e.g., [Rayner *et al.*, 2000; Bos *et al.*, 2003]. In this paper, we combine these ideas in a spoken language interpretation system for a robot. Our contributions are: (1) a probabilistic formalism for the selection of the interpretation with the highest posterior probability that integrates spoken and sensory input; and (2) a probabilistically-motivated interpretation process that takes into account alternatives derived from the user’s utterance and expectations obtained from the context.

This paper is organized as follows. In Section 2 we present an overview of the interpretation process, and in Section 3 we calculate the probability of an interpretation. Section 4 describes our interpretation process in more detail, followed by preliminary results in Section 5. Related work is discussed in Section 6, followed by concluding remarks.

## 2 Overview of the Interpretation Process

The procedure for generating interpretations receives as input a speech wave obtained from a user’s utterance, and generates one or more frame-based interpretations. Each interpretation is assigned a probability based on how well it fits the speech wave and how reasonable it is in the current context.

The interpretation of an utterance is performed by a multi-stage search process, where the best candidates generated at each stage are used as the basis for the next stage, and the probability of a candidate generated at a particular stage contributes to the probabilities of its “children” generated at the next stage. Figure 3(a) shows the interpretation process, and Figure 3(b) illustrates an example of a single search sequence for the sentence “Get the blue mug”.

Our architecture investigates the use of general syntactic information and semantic information for as long as possible, and the incorporation of domain-related information only in the final stages of the interpretation process. Specifically, no domain knowledge is used for the first three stages of the interpretation process (in fact, general off-the-shelf tools are used for the first two stages). In addition, domain knowledge is gradually incorporated in an interpretation, starting with abstract features that are relevant to our domain, then matching domain actions to verbs, and finally matching domain entities to the other elements of an interpretation. The expectation from our approach is that it will be able to process relatively unconstrained utterances by casual users, at least

up to a point, and provide the opportunity for the system to learn additional forms of expression and mappings between linguistic entities and domain actions and objects. Further, our architecture clearly delineates the portions of the system that should be modified when porting it to a different domain.

In the first stage of the interpretation process, the system activates an Automatic Speech Recognizer (ASR), which generates candidate sequences of words (*Text*). A probabilistic parser then generates *Parse Trees* for the most promising word sequences. The best Parse Trees are used next to construct *Parse Frames*. Each element of a Parse Frame is assigned a semantic role, e.g., Action, Agent, Patient and Recipient, and points to a sub-tree in the parent Parse Tree. The feature extraction stage applies rules to extract semantic features from the Parse Frame. Examples of these features for nouns are: definite/indefinite, colour and size (this stage collapses several Parse Frames into one Feature Frame). In the next stage, the system builds *Action Frames* composed of domain actions and constraints that match the elements in each Feature Frame. For instance, the word “get” maps to several domain actions, e.g., *fetch\_deliver* and *fetch\_person*. Each of these mappings has its own constraints that apply to the Agent, Patient, etc in the frame, e.g., *fetch\_deliver* requires the Patient to be a movable thing, and *fetch\_person* requires the Patient to be a person. In the final stage of the interpretation process, the entities and constraints in the Action Frame are unified with actual entities in the domain, yielding a *Unified Frame*.

Our current prototype generates a ranked list of interpretations. This list will be used to determine a course of action as follows. If one of the Unified Frames generated by DORIS is a clear winner, i.e., its posterior probability exceeds an “acceptable” threshold and is significantly higher than that of the other interpretations, then it will be selected by the system. Otherwise, DORIS will ask a clarification question. Further, the probabilities calculated at each stage of the interpretation process will be used to decide whether a clarification question should be asked at that stage.

### 2.1 Anytime algorithm

A search algorithm generates a *search graph* by *expanding* candidate options. This is done by means of an selection-expansion cycle that is repeated until all the options have been considered (full expansion) or until the system runs out of resources, e.g., time or memory. The selection step determines which option is to be expanded next, and the expansion step generates the children of that option (e.g., the children of a Feature Frame are Action Frames).

Since full expansion is often not feasible, we apply an interpretation process based on the anytime algorithm described in [George and Zukerman, 2004], which produces a plausible interpretation after a few seconds (Section 5). Anytime algorithms produce reasonable results at any point in time [Dean and Boddy, 1988]. These results generally improve as time elapses, typically exhibiting an asymptotic behaviour. Thus, our interpretation algorithm can act on a “good” interpretation produced after a few seconds, but can continue processing in the background, and “change its mind” if a better interpretation is discovered later.

Our anytime algorithm differs from normal search algorithms in that it expands a selected option one child at a time,

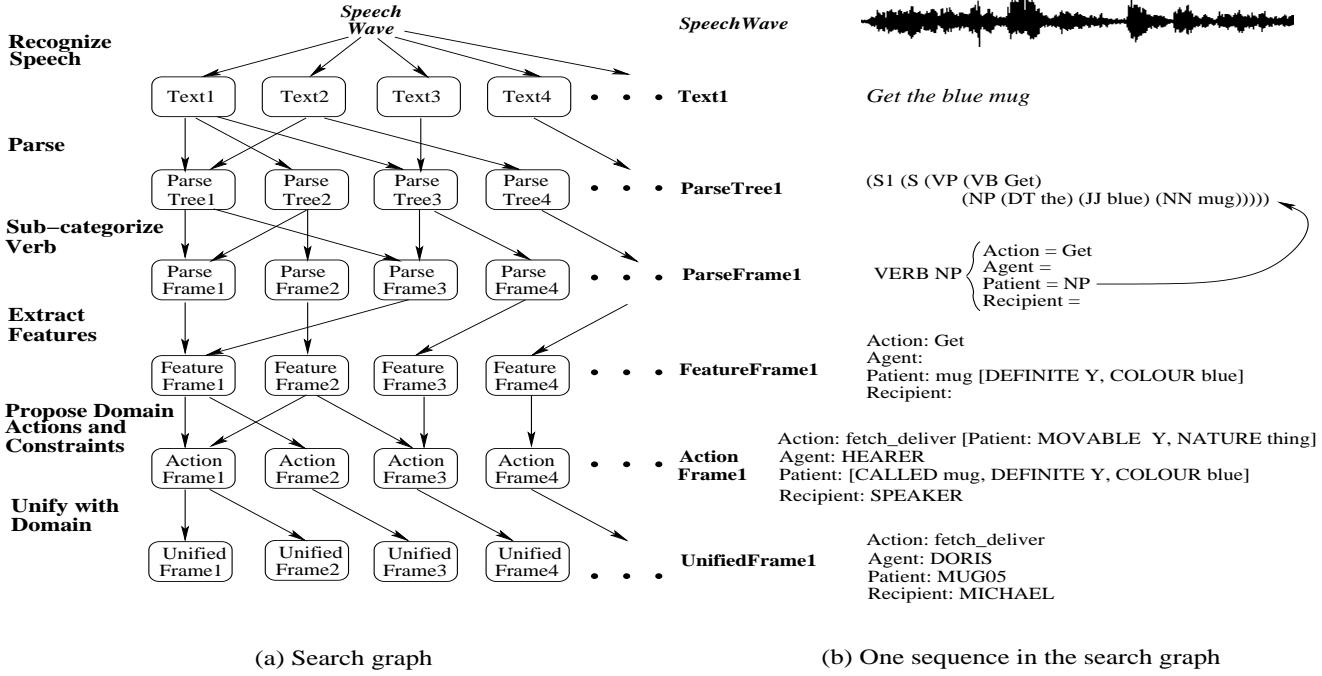


Figure 3: Flow of information in the interpretation process

rather than generating all its children at once, e.g., when it expands a Feature Frame, it generates a single Action Frame. The selection process then determines which option to expand next. This could be the same option that was expanded before (thereby generating another child), or a different option. During the selection process, our algorithm selects a level to expand next, and an option within this level.

- Selecting a level – preference is given to lower levels in the search graph by applying the following “coin-throwing” procedure. Assign to level  $i$  ( $= 6 = \text{Action Frame}$ ) a probability of 0.5 of being selected. If level  $i$  is not selected, set  $i \leftarrow i - 1$  and repeat this process. This policy produces a search tree that is roughly binary, i.e., each level is about twice the size of the previous level. In the future, we will investigate level-selection policies that are sensitive to the quality of the output at each level.
- Selecting an option within a level – here we apply the coin-throwing procedure with a probability  $\frac{1}{3}$  of selecting the most promising candidate. The promise of a candidate is a function of the probabilities of its previously generated children moderated by their recency. That is, a candidate that has produced good children recently is considered more promising than one that has produced good children followed by bad ones. Further, a childless candidate is deemed to have the most promise. This means that our algorithm tries to expand each option at least once.

### 3 Probabilistic Framework

As stated in Section 1, DORIS selects the interpretation with the highest posterior probability in light of the given speech wave and the conversational context. Following Miller *et al.* [1996], we are looking for  $\text{UnfdFrm}_{\text{Best}}$  such that

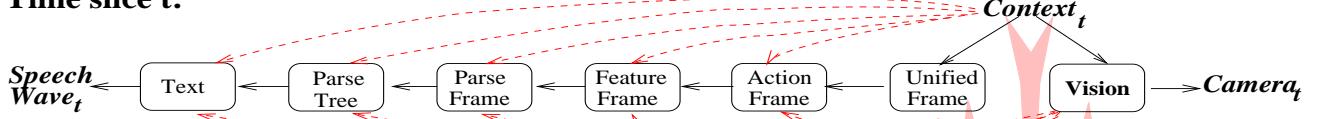
$$\text{UnfdFrm}_{\text{Best}} = \operatorname{argmax}_i \Pr(\text{UnfdFrm}_i | \text{Speech}, \text{Context})$$

The structures generated at each stage of the interpretation process are assigned probabilities in light of the available information, which may be evidential or causal. The evidence comes from the user’s spoken input (speech wave), and the cause comes from the conversational context. This context depends on previous input.<sup>1</sup> The Unified Frames obtained from the interpretation of the user’s previous utterance, the items identified by the vision system, and the information that existed previously in the context (in the future, the context will also be updated by the response generated by the system in the last conversational turn). We represent these probabilistic and temporal relations by means of a structure inspired by Dynamic Bayesian Networks (DBNs) – a formalism which incorporates temporal reasoning into a Bayesian Network [Dean and Wellman, 1991; Dagum *et al.*, 1992].

Figure 4 depicts two time slices of such a DBN. The arrows in the DBN represent the direction of causality in the conceptual model, and the direction of the calculations depends on the observed evidence. That is, the message intended by the user (which is hopefully the interpretation inferred by the system) causes the speech wave produced by the user, which in turn provides evidence for inferring the interpretation in question. Thus, the interpretation, i.e., the Unified Frame, is influenced evidentially by “upwards” propagation from the speech wave, but it is also affected causally by “downwards” propagation from the contextual information. In addition, the context at time  $t + 1$  depends on the context at time  $t$ , the Unified Frames obtained at time  $t$ , and the output of the vision system at time  $t$ . These influences, depicted with thick, grey arrows in Figure 4, are calculated algorithmically (rather than probabilistically). In principle, the context and vision

<sup>1</sup>At present we consider only speech and vision, and we assume that the system and the user can see the same things.

## Time slice t:



## Time slice t+1:

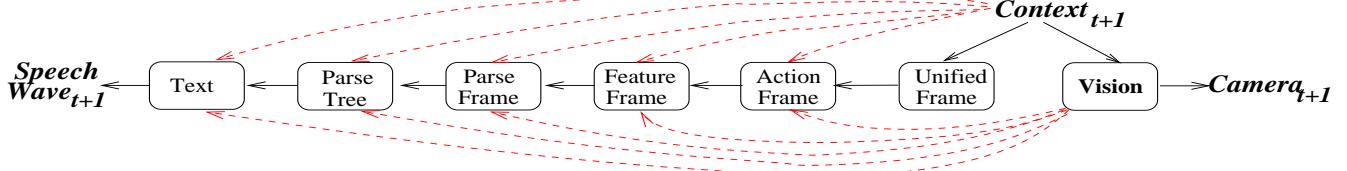


Figure 4: Two time slices of a DBN depicting causal and evidential influences during the interpretation process

also affect the other structures in the DBN (these effects are depicted with grey, dashed links in Figure 4). For example, we are likely to use a personal pronoun or a definite article to refer to a recently discussed object, and a demonstrative pronoun to refer to an object in our line of vision. However, these influences are not considered in the current implementation.

### 3.1 Calculating the probabilities of interpretations

We approximate belief propagation for each time slice of our DBN as follows.

**Evidence only.** This part of our calculation is similar to that described in [Miller *et al.*, 1996]. Equation 1 propagates evidence through the front of the DBN (up to Action Frame).

$$\Pr(\text{ActnFrm}|\text{Speech}) = \quad (1)$$

$$\sum_{\text{Txt}, \text{PrsTr}, \text{PrsFrm}, \text{FtrFrm}} \Pr(\text{ActnFrm}, \text{FtrFrm}, \text{ParseFrm}, \text{ParseTr}, \text{Text}|\text{Speech})$$

By assuming conditional independence between each stage and all but its directly preceding stage (e.g., a Feature Frame depends only on the Parse Frames from which it is generated), we obtain Equation 2.

$$\Pr(\text{ActnFrm}|\text{Speech}) = \quad (2)$$

$$\sum_{\text{Txt}, \text{PrsTr}, \text{PrsFrm}, \text{FtrFrm}} \left\{ \Pr(\text{ActnFrm}|\text{FtrFrm}) \times \Pr(\text{FtrFrm}|\text{ParseFrm}) \right. \\ \left. \times \Pr(\text{ParseFrm}|\text{ParseTr}) \times \Pr(\text{ParseTr}|\text{Text}) \right. \\ \left. \times \Pr(\text{Text}|\text{Speech}) \right\}$$

The probabilities in Equation 2 are obtained as follows.  $\Pr(\text{Text}|\text{Speech})$  is calculated from a score returned by the ASR.  $\Pr(\text{ParseTr}|\text{Text})$  is returned by the probabilistic parser. In the current implementation, the rest of the probabilities in Equation 2 depend only on the number of alternatives generated from the “parent” structure. In the future, we will take into account corpus-based information [Miller *et al.*, 1996], and consider the influence of the conversational context.

**Cause and Evidence.** The probability of a Unified Frame in light of causal and evidential information is given by Equation 3 (given a Unified Frame, we assume conditional inde-

pendence between an Action Frame and the Context).

$$\sum_{\text{ActnFrm}} \Pr(\text{UnfdFrm}|\text{ActnFrm}, \text{Context}) = \quad (3) \\ \alpha \sum_{\text{ActnFrm}} \{ \Pr(\text{ActnFrm}|\text{UnfdFrm}) \times \Pr(\text{UnfdFrm}|\text{Context}) \}$$

where  $\alpha$  is a normalizing constant.

We rewrite the first factor to reflect how probabilities are obtained from the interpretation process, yielding

$$\sum_{\text{ActnFrm}} \Pr(\text{UnfdFrm}|\text{ActnFrm}, \text{Context}) = \quad (4) \\ \beta \sum_{\text{ActnFrm}} \left\{ \Pr(\text{ActnFrm}) \times \Pr(\text{UnfdFrm}|\text{ActnFrm}) \right. \\ \left. \times \Pr(\text{UnfdFrm}|\text{Context}) \right\}$$

where  $\beta$  is a normalizing constant.

The probabilities in Equation 4 are obtained as follows.

- $\Pr(\text{ActnFrm}) = \sum_{\text{Speech}} \Pr(\text{ActnFrm}|\text{Speech}) \Pr(\text{Speech})$ . Since there is only one speech wave with  $\Pr(\text{Speech}) = 1$ ,  $\Pr(\text{ActnFrm}) = \Pr(\text{ActnFrm}|\text{Speech})$ .
- To calculate  $\Pr(\text{UnfdFrm}|\text{ActnFrm})$ , at present we make a simplifying assumption whereby we assume conditional independence between each element in  $\text{UnfdFrm}$  and the other elements in  $\text{UnfdFrm}$ , given the corresponding element in  $\text{ActnFrm}$ . This assumption yields

$$\Pr(\text{UnfdFrm}|\text{ActnFrm}) = \quad (5) \\ \prod_{i=1}^n \Pr(\text{element } i \text{ in } \text{UnfdFrm} | \text{element } i \text{ in } \text{ActnFrm})$$

where  $n$  is the number of elements in a frame.

This assumption does not hold in cases where the value of one element in a frame influences the value of other elements, e.g., “Give Sarah her book”. However, in many cases it is acceptable, since when generating an interpretation, our system propagates the constraints between the elements in a frame (Section 4). For instance, the constraints [Patient:MOVABLE Y, NATURE thing] in ActionFrame1 in Figure 3(b) are propagated to the Patient. As a result, each element in the Action Frame contains all the information required to determine the extent

ORIGINAL: get the cup	
ASR OUTPUT:	Total
get/-5 the/-2 cup/2	-5
debt/-6 the/-5 cup/2	-9
get/-6 their/-6 cup/2	-10
get/-5 it/-2 there/-5 cup/2	-10
get/-5 the/-5 cup/0	-10
.	.

Figure 5: Sample output of ViaVoice

to which it matches a candidate entity from the domain (and calculate the probability of the match to the corresponding element in the Unified Frame).

- Similarly, to calculate  $\Pr(\text{UnfdFrm}|\text{Context})$  we assume conditional independence between each element in  $\text{UnfdFrm}$  and the other elements in  $\text{UnfdFrm}$ , given the value of the corresponding element in  $\text{Context}$ .

$$\begin{aligned} \Pr(\text{UnfdFrm}|\text{Context}) &= \\ &\prod_{i=1}^n \Pr(\text{element } i \text{ in } \text{UnfdFrm} | \text{element } i \text{ in } \text{Context}) \end{aligned} \quad (6)$$

Our calculation of the probability of a domain entity in the Unified Frame, e.g., MUG05, from the context is based on the salience (i.e., level of activation) of this entity. The technique described in [Zukerman *et al.*, 2003] calculates salience from the previous references to an entity, the previous times it was seen, and its similarity to other mentioned or seen entities. We propose to adapt this technique to also take into account visual salience.

## 4 The Stages of the Interpretation Process

In this section, we flesh out some of the detail for each stage of the interpretation process outlined in Section 2.

### 4.1 Recognize spoken utterances $\Rightarrow$ Text

We employ ViaVoice (<http://www-306.ibm.com/software/voice/viavoice/>) as the speech recognition component. ViaVoice takes as input a speech wave uttered by a user, and generates several alternative word sequences, where each word has a score between -100 and +100 that indicates ViaVoice’s certainty. Figure 5 illustrates the output generated by ViaVoice for the sentence “Get the cup”. As seen in this Figure, the ASR is rather uncertain about its output, as the best word sequence has a total score of -5.

### 4.2 Parse word sequences $\Rightarrow$ Parse Tree

The word sequences generated by ViaVoice are parsed using Charniak’s probabilistic parser (<ftp://ftp.cs.brown.edu/pub/nlpars>). For each word sequence, the parser produces a set of parse trees ranked according to their probability. Figure 6 shows the five top parse trees generated by Charniak’s parser for the word sequence “get the cup”, together with the negative log of their probability.<sup>2</sup>

<sup>2</sup>We use negative logs of probabilities, rather than probabilities, to make the calculations more tractable. Note that the higher the probability, the lower its negative log.

ASR OUTPUT: get the cup	-log Pr
PARSER OUTPUT:	28.4608
(S1 (S (VP (VB get) (NP (DT the) (NN cup))))))	29.2191
(S1 (VP (VB get) (NP (DT the) (NN cup))))	31.7986
(S1 (S (VP (VBP get) (NP (DT the) (NN cup))))))	32.0519
(S1 (VP (VB get) (S (NP (DT the) (NN cup))))))	33.2125
.	.

Figure 6: Sample output of Charniak’s probabilistic parser

### 4.3 Sub-categorize the verb $\Rightarrow$ Parse Frame

This stage takes as input a Parse Tree, and generates a small set of Parse Frames with pointers to sub-trees of the Parse Tree. This is done by applying the following procedure.

- Determine the main verb in the Parse Tree, and extract from a lexicon the sub-categorization patterns for this verb. Each sub-categorization pattern has a set of semantic-category assignments that fit the pattern. For instance, the patterns for the verb “get” are {[NP PP], [NP NP], [NP]}, and the semantic-category assignments for [NP NP] are: [Patient=NP1; Recipient=NP2] and [Patient=NP2; Recipient=NP1].<sup>3</sup>
- Retain only the verb sub-category patterns that match the dependencies in the VP sub-tree. Create a Parse Frame for each semantic-category assignment in the retained patterns, and store a pointer from assignment to the appropriate sub-tree in the Parse Tree. For instance, the request “Get NP[Sarah] NP[a mug]” yields two Parse Frames for the sub-category pattern [NP NP] with the following values for Patient and Recipient:

...Patient=NP[a mug];Recipient=NP[Sarah]  
...Patient=NP[Sarah];Recipient=NP[a mug]

At present, our rules for selecting semantic-category assignments are rather coarse, and do not use syntactic information (e.g., about the preposition in a PP) to discredit (reduce the probability of) invalid alternatives. As a result, such alternatives are retained longer than strictly necessary, and are identified only in the last step of the interpretation process (using feature-unification and semantic-constraint propagation, Section 4.6). This is an efficiency issue, which we intend to address in the near future by considering more precise rules for making semantic-category assignments.

### 4.4 Extract semantic features $\Rightarrow$ Feature frame

This stage receives as input a Parse Frame, and generates a Feature Frame containing a set of features derived from the sub-trees in the Parse Frame. The process for obtaining features distinguishes between entities and actions as follows.

- Entities** – Our system examines the sub-tree pointed to by the Patient and Recipient in a Parse Frame in order to extract features from the structures attached to

<sup>3</sup>In the current implementation, the lexicon contains 50 verbs, and the patterns are composed of NPs and PPs only. Additional detail will be added in the near future.

the head noun in this sub-tree. We distinguish between specific features that are important in our domain and generic features. The specific features, which include whether the noun is definite, as well as colour and size, are recorded separately in the Feature Frame. The generic features are treated as words and placed in a generic category. For example, the NP “DT[the] ADJP[old green] mug” has the features {DEFINITE Y, COLOUR green, GENERIC [old]}.

- **Actions** – In contrast to entities, all the structures in a VP that have not been matched with semantic sub-categories are attached to the Action (verb) as generic features.

This process yields a many-to-one mapping, as several Parse Frames may produce the same Feature Frame. For instance, the sentence “Get NP[the big red ball]” yields several Parse Frames, two of which have the following Patients:  
P1: NP[DT the][ADJP[JJ big][JJ red]][NNS ball]  
P2: NP[DT the] [JJ big][JJ red] [NNS ball]  
Both of these patients yield the following Patient component, and hence a single Feature Frame:

```
[Patient:ball,DEFINITE Y,COLOR red,SIZE big]
```

#### 4.5 Propose domain actions and constraints ⇒ Action Frame

This stage receives as input a Feature Frame, and returns one or more Action Frames. For each action and sub-categorization pattern in the FeatureFrame, we extract one or more domain actions from a verb lexicon (which at present contains 50 entries, most with several senses, mapped to 31 domain actions). These actions are associated with constraints for the elements in the resultant Action Frame. For example, the verb “find” yields the domain actions `locate_inform` and `fetch_deliver`, which have the following constraints for the Patient:

```
locate_inform:[Patient:MOVABLE N]
locate_inform:[Patient:NATURE person]
fetch_deliver:[Patient:MOVABLE Y,NATURE thing]
locate_inform finds non-movable entities or people, and then informs the person who made the request of their whereabouts, e.g., “find the kitchen” and “find Ingrid”; while fetch_deliver finds and retrieves movable objects, e.g., “find a cup”. Each domain action yields an Action Frame whose elements include assigned constraints. We distinguish between three types of constraints on the basis of how they are processed.
```

- Constraints for non-action elements, e.g., the Patient must be movable. These constraints are propagated to the elements in question in order to match them with entities in the domain (Section 4.6).
- Constraints for the action, e.g., the action must be done quickly. These constraints will be used by a planner when determining a response (the planner will be implemented in the near future).
- Constraints relating to implied information, i.e., information that has been omitted by the speaker, and must be filled in to support the generation of a Unified Frame from the Action Frame, e.g., the Recipient must be the speaker.

This process yields mainly a one-to-many mapping. That is, a Feature Frame normally produces several Action Frames.

#### 4.6 Unify Action Frames with domain entities ⇒ Unified Frame

This stage receives as input an Action Frame and contextual information, and returns one or more Unified Frames, where the head nouns in the Action Frame have been replaced with objects in the physical world. The current implementation represents 27 entities in the physical world, which are assumed to be correctly recognized with certainty (our architecture provides a representation of uncertainty regarding visual identification of real-world objects, but we have not yet incorporated this uncertainty into our probabilistic framework).

As seen in Section 3, the probability of each Unified Frame, which is calculated using Equation 4, depends on the salience of its physical objects according to the context, how well these objects match the constraints and features in the given Action Frame, and the probability of the Action Frame. In this section, we focus on the second component.

Each head noun in the Action Frame is compared with the objects in the real world. The better the match between a physical object and the constraints and features of the head noun, the higher the probability that this object was intended by the head noun. Unified Frames are then constructed from combinations of high-probability objects. As seen in Equation 5, the probability of a Unified Frame given an Action Frame is the product of the probabilities of these objects.

The calculation of the goodness of a match depends on the constraints and the type of the features under consideration. Features with many values, such as COLOUR and SIZE, yield a wide range of probabilities, while constraints, such as NATURE person, yield extreme probability values. For example, if the head noun of the Patient of an Action Frame had the features [CALLED cup, COLOUR blue], and the physical world had several cups, of which CUP01 is blue and CUP05 is aqua, the system would produce several Unified Frames, of which one would have [Patient CUP01] and another [Patient CUP05]. The probability of the CUP01 interpretation would be higher than that of the CUP05 interpretation, the probabilities of interpretations with red and yellow cups would be much lower, and the probability of an interpretation with a blue lamp would be close to zero.

## 5 Results

In this section, we describe preliminary results obtained by DORIS for a virtual domain that consists of 27 domain entities (things, people and locations) and 31 domain actions.

In our experiments, we imposed no restrictions on the number of expansions performed by the interpretation stages (the off-the-shelf systems, ViaVoice and Charniak’s parser, have their own internal limits). The number of alternatives generated by these stages is determined by our anytime algorithm (Section 2.1).

We ran 11 simple commands which were variants of “cook the pizza”, “feed the cat”, “find Ingrid”, “get the blue cup”, “find the cup” and “open the cupboard”. For all inputs, except “cook the pizza”, the top-ranked interpretation was plausible. Ambiguities arose when a few domain objects matched the input and were equally salient, e.g., “feed the cat” yields equiprobable interpretations where cat0 or cat1 are to be fed. “Cook the pizza” was problematic because none of the options generated by ViaVoice were correct: the top-ranked

option was “look for pizza”, and the option that was closest to the spoken input, viz “cook for pizza”, had a much lower score.

The last four stages of the interpretation process (without ViaVoice and Charniak’s parser) took less than 1 second in total to generate each of the top-ranked interpretations; the average time for producing the top-ranked interpretation (over 11 runs) was 0.15 seconds. These times are consistent for the top five interpretations generated for all inputs except for “cook the pizza”, where the correct interpretation (ranked fourth) was found in 4.54 seconds (this is because the “cook for pizza” ViaVoice output was inspected late, in accordance with its ranking).

Let us now consider three examples that illustrate the effect of constraints on the domain action and domain entity in an interpretation, and the influence of conversational context on the preferred interpretation. The inputs for these examples are: “find Ingrid”, “find the cup” and “open the cupboard” (spoken by Michael). As shown in Section 4.5, the verb “find” can be mapped to `locate_inform`, which requires non-movable entities or people, and `fetch_deliver`, which requires movable objects. The most probable Unified Frame for “find Ingrid” was  $UF_1 = \{\text{Act:locate\_inform, Pat:ingrid, Rec:michael}\}$  with  $-\log \Pr(UF_1) = 70.9926$ . For “find the cup”, DORIS generated two best Unified Frames, which correspond to the two cups known to DORIS:

$$UF_2 = \{\text{Act:fetch\_deliver, Pat:cup0, Rec:michael}\}$$

$$UF_3 = \{\text{Act:fetch\_deliver, Pat:cup1, Rec:michael}\}$$

with  $-\log \Pr(UF_i) = 73.4494$  for  $i = 2, 3$ .

To illustrate the influence of the conversational context on the preferred interpretation, we repeated the “find the cup” request in a context that had been manually modified to simulate a recent mention of `cup0` (`cup0` was five times more salient than `cup1`). This led to an increase in the probability of the `cup0` interpretation to  $-\log \Pr(UF_2) = 71.1274$ , thereby making it the preferred interpretation, while the probability of the `cup1` interpretation remained unchanged.<sup>4</sup>

A similar outcome was obtained for the “open the cupboard” example. The option preferred by ViaVoice, “open the cabinet”, yielded the winning interpretation when both the cupboard and the cabinet were equally salient. When the cupboard became more salient than the cabinet, then “open the cupboard” won. Table 1 shows several outputs produced by ViaVoice for this example (left-hand side), together with the processing stage where the interpretation was discredited (its probability was reduced), and the reason for this outcome. Both “open the cup” and “open the couplet” failed when building a Unified Frame – the former due to a constraint failure (cups are not “openable”), and the latter because DORIS does not know about “couplet” domain objects. “Plan the cupboard” failed a bit earlier, during Action Frame construction, because DORIS has no domain action for “plan”. Clearly, these last two failures would not have occurred if ViaVoice and the parser used only the restricted vocabulary corresponding to the domain actions and entities known to

<sup>4</sup>If we generated a complete set of interpretations, their probabilities would be normalized, and the probability of the `cup1` interpretation would go down. However, the generation of an exhaustive set of interpretations is generally not feasible.

Table 1: Processing stages for “open the cupboard”

ViaVoice output	Stage where discredited (Reason)
<i>in the cupboard</i>	Sub-categorize verb (no verb)
<i>pin the cupboard</i>	—
<i>plan the cupboard</i>	Gen. action frame (no domain action)
<i>open the cabinet</i>	—
<i>open the cupboard</i>	—
<i>open the cup</i>	Gen. unified frame (constraint failed)
<i>open the couplet</i>	Gen. unified frame (no domain object)

DORIS (an approach adopted in [Matsui *et al.*, 1999]). However, this vocabulary mismatch gives DORIS the potential to learn new domain actions and objects.

## 6 Related Research

We have extended the maximum posterior probability approach generally used for speech recognition [McTear, 2002; Young, 2002] to all the stages of the spoken language understanding process. Such an extension was also performed by [Miller *et al.*, 1996; He and Young, 2003] for the interpretation of utterances from the ATIS corpus. However, both Miller *et al.* and He and Young used a semantic grammar for parsing, while we use syntactic information and general semantic information for as long as possible, incorporating domain-related information only in the later stages of the interpretation process. Knight *et al.* [2001] compared the performance of a grammar-based dialogue system with that of a system based on a statistical language model and a robust phrase-spotting grammar. Their results show that the latter perform better for relatively unconstrained utterances by users who are not familiar with the system. This finding supports our general approach, which relies on a statistical language model.

As done by several researchers in spoken language systems in general, and robot-based systems in particular (e.g., [Matsui *et al.*, 1999; Rayner *et al.*, 2000; Lemon *et al.*, 2002; Bos *et al.*, 2003]), we also take into account expectations from contextual information during language interpretation. Matsui *et al.* used contextual information to constrain the alternatives considered by the ASR early in the interpretation process. This allows their system to process expected utterances efficiently, but makes it difficult to interpret unexpected utterances. Rayner *et al.* used contextual information to produce different interpretations from contextually available candidates, and to resolve anaphora and ellipsis. Our architecture resembles that described in [Rayner *et al.*, 2000] in its successively deeper levels of representation, and its consideration of several options at each level. However, we provide a probabilistic framework for the selection of interpretations, while their selection process is based on diagnostic messages produced during the interpretation process. Lemon *et al.* used a Bayesian Network (BN) that takes into account activity and recency as well as the user’s and the system’s previous dialogue move for anaphora resolution. However, this BN is not integrated with the rest of their language processing system. Finally, Bos *et al.* developed a dialogue system for a mobile robot called Godot, which like DORIS, understands natural descriptions, such as “the kitchen”, and takes context into account. However, unlike DORIS, Godot’s language in-

terpretation process uses a logic-based framework, and employs formal proofs for conflict resolution.

The probabilistic formalism presented in this paper uses a DBN-like structure to integrate context-based expectations with alternatives obtained from spoken utterances. Further, our formalism is designed to incorporate information from other channels, e.g., touch and vision, into the interpretation process, and to enable the interpretations obtained from different channels to influence each other. Our work resembles that of Horvitz and Paek [1999; 2000] in these two respects. However, Horvitz and Paek focus on higher level informational goals than those addressed in this paper, using a single output produced by a parser as linguistic evidence for their goal recognition system. Further, they consult additional information channels, e.g., vision, only after the spoken input has yielded insufficient information, while our approach considers all channels in tandem. Finally, our probabilistic approach supports the future integration of our language interpretation system into a utility-based decision procedure, such as that described in [Horvitz and Paek, 1999; Paek and Horvitz, 2000].

## 7 Conclusion

We have offered (1) a probabilistic formalism for the selection of the interpretation with the highest posterior probability, and (2) a multi-layered interpretation process that takes into account alternatives derived from a user's utterance and expectations obtained from the context. Our probabilistic formulation and our interpretation process support the integration of sensory and spoken input. Further, our interpretation process employs general syntactic information and semantic information for as long as possible, incorporating domain-related information only in its final stages.

## Acknowledgments

This research was supported in part by the ARC Centre for Perceptive and Intelligent Machines in Complex Environments. The authors thank Eugene Charniak for his modifications to his probabilistic parser, and Charles Prosser for his assistance in extracting multiple texts from ViaVoice.

## References

- [Bos *et al.*, 2003] J. Bos, E. Klein, and T. Oka. Meaningful conversation with a mobile robot. In *EACL10 – Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 71–74, Budapest, Hungary, 2003.
- [Dagum *et al.*, 1992] P. Dagum, A. Galper, and E. Horvitz. Dynamic network models for forecasting. In *UAI92 – Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 41–48, Stanford, California, 1992.
- [Dean and Boddy, 1988] T. Dean and M.S. Boddy. An analysis of time-dependent planning. In *AAAI-88 – Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, St. Paul, Minnesota, 1988.
- [Dean and Wellman, 1991] T. Dean and M.P. Wellman. *Planning and control*. Morgan Kaufmann Publishers, San Mateo, California, 1991.
- [George and Zukerman, 2004] S. George and I. Zukerman. An anytime algorithm for interpreting arguments. In *PRI-CAI2004 – Proceedings of the Eighth Pacific Rim International Conference on Artificial Intelligence*, pages 311–321, Auckland, New Zealand, 2004.
- [He and Young, 2003] Y. He and S. Young. A data-driven spoken language understanding system. In *ASRU'03 – Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, St. Thomas, US Virgin Islands, 2003.
- [Horvitz and Paek, 1999] E. Horvitz and T. Paek. A computational architecture for conversation. In *UM99 – Proceedings of the Seventh International Conference on User Modeling*, pages 201–210, Banff, Canada, 1999.
- [Knight *et al.*, 2001] S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling, and I. Lewin. Comparing grammar-based and robust approaches to speech understanding: A case study. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, 2001.
- [Lemon *et al.*, 2002] O. Lemon, P. Parikh, and S. Peters. Probabilistic dialogue modeling. In *SIGDIAL02 – Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pages 125–128, Philadelphia, Pennsylvania, 2002.
- [Matsui *et al.*, 1999] T. Matsui, H. Asoh, J. Fry, Y. Motomura, F. Asano, T. Kurita, I. Hara, and N. Otsu. Integrated natural spoken dialogue system of Jijo-2 mobile robot for office services. In *AAAI99 – Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 621–627, Orlando, Florida, 1999.
- [McTear, 2002] M.F. McTear. Spoken dialogue technology: Enabling the conversational user interface. *ACM Computing Surveys*, 34(1):90–169, 2002.
- [Miller *et al.*, 1996] S. Miller, D. Stallard, R. Bobrow, and R. Schwartz. A fully statistical approach to natural language interfaces. In *ACL96 – Proceedings of the 34th Conference of the Association for Computational Linguistics*, pages 55–61, Santa Cruz, California, 1996.
- [Paek and Horvitz, 2000] T. Paek and E. Horvitz. Conversation as action under uncertainty. In *UAI-2000 – Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 455–464, Stanford, California, 2000.
- [Rayner *et al.*, 2000] M. Rayner, B.A. Hockey, and F. James. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 112–118, Seattle, Washington, 2000.
- [Young, 2002] S. Young. Talking to machines (statistically speaking). In *ICSLP'02 – Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 113–120, Denver, Colorado, 2002.
- [Zukerman *et al.*, 2003] I. Zukerman, S. George, and M. George. Incorporating a user model into an information theoretic framework for argument interpretation. In *UM03 – Proceedings of the Ninth International Conference on User Modeling*, pages 106–116, Johnstown, Pennsylvania, 2003.

# Planning clarification questions to resolve ambiguous references to objects

Jeremy L Wyatt

Intelligent Robotics Laboratory

School of Computer Science

University of Birmingham

Birmingham, UK, B15 2TT

jeremy.wyatt@britishlibrary.net

## Abstract

Our aim is to design robots that can have task directed conversations with humans about objects in a table top scene. One of the pre-requisites is that the robot is able to correctly identify the object to which another speaker refers. This is not trivial as human references to objects are often ambiguous, and rely on contextual information from the scene, the task, or the dialogue to resolve the reference. This paper describes work in progress on building a robot system able to plan the content of clarifying questions that when answered provide the robot with enough information to resolve ambiguous references. It describes an algorithm that models the degree of uncertainty about the binding of a referent using a probability distribution. We use the visual salience of the object as a way to generate the prior distribution over candidate objects, which we call the belief state. Then we generate action models, for the effects of various clarifying questions, on the fly. Finally we evaluate the mean reduction in the entropy of the resulting belief states. The method can be seen as a form of prior-posterior analysis, or as one step look ahead in an information state Markov decision process. We are currently implementing the algorithm in a robot and discuss the issues we have encountered to date.

## 1 Introduction

Human-robot communication is an increasingly active field [Roy *et al.*, 2004; Sidner *et al.*, 2004; Sidner and Dzikovska, 2004; Oates *et al.*, 2000], with many challenging problems. One of the most basic abilities for a robot capable of conversing with a human about objects in a scene is the ability to bind the references made by a speaker to objects in the world. One of the problems of human dialogue is that references to objects are often linguistically underspecified. Because of this the robot may need to incorporate other information to resolve the reference. An example is the case below:

- H: “What is to the left of the red cup?”
- R: “Is it the large red cup?”
- H: “Yes.”



Figure 1: A scene with three red mugs, and two other objects.

- R: “There is a green ball to the left of that cup.”

Here the tutor has asked the robot about the identity of an object with a particular relationship to another object (a red cup) in the scene. First of all the question makes a reference to a red object, and indirectly (through the spatial relationship to this object) to the object of interest. Answering the question requires that the robot is capable of decoding this indirect reference, i.e. figuring out which object is the one to the left of the red cup. Here the reference involves an ambiguous reference to another object (a red cup) which is being used as a landmark. If the reference to the red cup is ambiguous (here there are two cups with significant areas of red, each with an object to their left), then the robot must take an action to resolve this ambiguity. This could involve checking to see if the human is pointing at the object, or it could involve asking a clarifying question as in the dialogue above. Where there are several possible clarifying actions we will require a system for generating clarifying questions and for deciding which one is most appropriate. In addition the system should be able to incorporate information from either language or vision.

In this paper we shall suppose that we have a vision system that is capable of producing a list of the objects in the scene, and their approximate positions on the ground plane. We will also assume that it is possible from these to generate a

scene graph, which we have already been able to do for small numbers of simple objects [Kruijff and Kelleher, 2005]. It is important to note that while this is possible for a very limited number of fixed objects, it is not possible for a wide variety of objects, or for cluttered or complex scenes. To make a real robot system scale, further mechanisms are necessary, particularly an attentional system. We return to this issue briefly in the final section. The rest of the paper is structured as follows. In section 2 we introduce a running example, in section 3 we describe the kind of clarifying actions we can take to resolve ambiguity. In section 4 we describe how we generate action models for the dialogue moves on the fly, and then in section 5 we describe the various metrics we use to evaluate them.

## 2 An example problem

Suppose that we have a vision system that is capable of building a list of objects together with their attributes. The attributes we can reasonably expect to get from our existing vision system are as follows:

```
Object      :category
             :colour
             :projective relations
             :proximity relations
             :position in image plane
             :position on ground plane
```

In order to extract the proximity relations we use a potential field model to capture the notion of nearness between objects [Kruijff and Kelleher, 2005]. Using this model we can extract the qualitative relationship *near* from the geometric locations of the objects in a way that is contextually sensitive. In Figure 2, for example we might say that object  $o_8$  is near  $o_2$ , but this will depend on how far away the objects are, their relative size, and where other objects are relative to them. To model projective relationships, such as *left of*, *right of*, *behind* and *in front of* we create simple tessellations of the space centred on the objects themselves. At the moment we only extract the relationships for the robot's own frame of reference, but given a geometric model it is possible to extend this to other frames of reference, i.e. speaker or object based.

We also have a discourse context, which consists of an information state and some additional structures. These are based around a set of logical forms, each of which represents the content of an utterance. Each logical form will contain one or more discourse referents. These include referents that are variables which can be bound to physical objects in the scene. A significant part of the task of integrating the visual and linguistically derived representations is to decide which object to bind to which discourse referent. We use a couple of running examples. In the first suppose the human utters the question: ‘‘What colour is the mug?’’ where there are objects arranged on the table as shown in Figure 2.

To answer this question the discourse manager needs to create a goal. This goal is to reach an information state where the human agent believes the mug to have a certain colour, and where that belief is correct. To satisfy this goal the robot will have to make an utterance stating the colour of the mug. In order to do this it needs to bind the referent to a physical

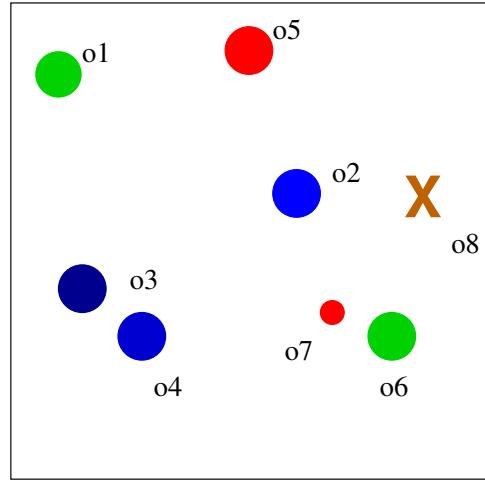


Figure 2: An overhead map of some objects on the table. Large circles represent mugs, the small circle represents a ball, the cross represents an object that has not been identified.

object. In this example there is a significant degree of referential ambiguity. In the second example, let's assume the same layout of objects on the table, but that the human instead says ‘‘Pick up the mug’’.

Let's also assume that when the discourse manager realises that a reference is ambiguous it makes a call to resolution routine *resolve(discourse\_referent, discourse\_context, scene\_graph)* which will plan and make an utterance to resolve the reference. We will now consider the actions that this procedure might consider.

## 3 Asking clarifying questions to resolve references

What are the different actions that the robot could take in order to resolve the reference above? In the first example (where we are asked for the colour of the mug) we will need a representation that captures the rule that it is not sensible to ask a question of the form ‘‘Is it the red mug?’’. Obviously, in the second example (‘‘Pick up the mug.’’) this question is fine. We don't discuss this problem further in this paper. Reasonable things the robot can do for example 1 include:

- Checking to see if the speaker is pointing at the object.
- Asking whether it is on the left or the right of the scene.
- Asking if it is at the back or the front of the scene.
- Asking if it has some projective or proximal relationship to some other object not in the set of objects that can be bound, e.g. ‘‘Is it next to the red ball?’’
- Asking the human to point at the mug.
- Asking the human the general question ‘‘Which mug is it?’’.

- Asking a question about which mug it is by referring to a mug using one of a number of other attributes e.g. ‘‘Is it the big mug?’’.

Reasonable things I can do for example 2 include all of the above, and questions about colour:

- ‘‘Is it the red mug?’’
- ‘‘What colour is the mug?’’
- ‘‘Is it the green mug next to the red ball?’’
- ‘‘Which mug is it?’’

In the following sections we consider the effects of questions about the attributes or relations of an object, of general questions (“which one is it?”), and of questions involving referring expressions. What we ideally want is a model of selection that while not enforcing the sorts of utterances that humans make, settles upon those utterances for good reasons. As an example we want a model that will typically not ask questions such as “Is it blue?”, as this is not a natural response, but would prefer questions such as “Is it the blue mug?” or “Which mug is it?”. We want this preference to arise out of sensible criteria (such as the cognitive load, or the expected gain in information), rather than by excluding certain classes of question from consideration. The question we should prefer will depend on the degree of ambiguity in the reference. Systems such as Ripley [Roy *et al.*, 2004] use a simple catch all strategy of asking “Which one is it?” whenever confronted with referential ambiguity. While such a strategy is quite effective humans use a wider range of expressions dependent on context and the degree of uncertainty. We want to be able to produce qualitatively similar behaviour from our model.

Our first problem is representing the degree of uncertainty about the reference, and using this to incorporate existing information from modalities other than language, e.g. vision. The objects in the set of possible referents have varying degrees of visual salience, large objects in the foreground are highly salient, whereas small objects in the background are not. There is a simple algorithm that lets us calculate the visual salience for an object in terms of its size on the image plane, and how central it is in the current view [Kelleher and van Genabith, 2004]. In that system the visual salience alone is used to resolve references to objects that have not previously appeared in the dialogue, but which appear in the visual scene. We extend this, by assuming that picking the most visually salient object that satisfies the reference may not be enough. In addition we want a mechanism that allows us to measure the degree of ambiguity after taking the salience into account. We propose that having obtained these saliences we normalise them and interpret them as probabilities in a prior:

$$p_i = \Pr(d_1 = o_i) = \frac{\text{salience}(o_i)}{\sum_{j \in D(d_1)} \text{salience}(o_j)} \quad (1)$$

where  $D(d_1)$  is the distractor set for the discourse referent  $d_1$ . We will refer to a probability distribution over the elements in the distractor set as a belief distribution. Let’s denote the belief distribution as follows:

$$B(d_1) = \{(o_1, p_1) \dots (o_i, p_i) \dots (o_7, p_7)\} \quad (2)$$

where  $p_i$  is the likelihood that the discourse referent binds to object  $o_i$ . The prior belief distribution  $B'$  is simply the distractor set augmented by the prior probability of each binding. We will sometimes refer to a belief distribution as a state. The distractor set itself is simply the set of objects in the object list (from the visual information) that match the properties of the discourse referent. Our discourse referent has a set of attributes and relations garnered from the utterance. We obtain this set by parsing with a combinatorial categorial grammar designed for talking about objects and their spatial relationships. For the utterance “Pick up the mug.” the attributes of the discourse referent can be simply represented in the same form as our object list supporting our scene graph.

`d1 :category=mug`

So initially our distractor set is:

$$D(d_1) = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\} \quad (3)$$

If  $D(d_1)$  had a single element then we would want `resolve` to terminate and return the only possible binding. If the set is empty, then we must return with that fact and look for a candidate object elsewhere. If there are several possible bindings, then we must ask clarifying questions to gain information in order to resolve the reference. Let’s denote asking a question as `ask(x, q)` where  $x$  is the agent asked and  $q$  is the question. For each question that we can ask we will assume that we can build an action model on the fly that effectively generates a set of possible new information states, in that it generates a set of possible new distractor sets after the question is answered.

## 4 Generating action models for dialogue moves on the fly

There are many algorithms for generating referring expressions that take an object and a distractor set for which we have a set of relations and properties and calculate an expression that will refer uniquely to that object if possible. So if we want to generate a question of the form ‘‘is  $d_1 = o_1?$ ’’ then we can call such an algorithm to generate a referring expression for  $o_1$ . In our algorithm we will, as part of the planning process, internally generate a referring expression for every object in the distractor set. We will therefore also assume that we are able to generate a call to an algorithm for generating referring expressions from the scene graph, `generate_reference(scene_graph)`. A reasonable algorithm that requires a scene graph and given this efficiently handles referring expressions involving properties and spatial relations is that of [Krahmer *et al.*, 2003]. These referring expressions will be the basis of questions of the form ‘‘Is it the ... ?’’.

Of course until we’ve called the algorithm we may not have any idea as to whether it is possible to generate such a referring expression given the information we have. Consider distinguishing between the two blue mugs on the left of Figure 2. Suppose that there is nothing in the scene graph that

enables us to generate a referring expression that uniquely identifies either of the two objects. If it is not possible to generate an unambiguous referring expression for an object then the least ambiguous referring expression can be returned, and a question can be considered which will ask about whether the attributes of the object match those returned (e.g. if we no information about spatial relations in the graph we could ask “Is it green?”.)

Once we have a method to generate possible questions, we must be able to model their effects and then evaluate them. Consider example 2: “Pick up the mug”. We may want to consider asking the clarification questions:

- “Is it the green mug on the left?”
- “What colour is the mug?”
- “Is it on the left”
- “Is it blue?”

Each one of these will move us to a new belief distribution  $B''$  (or equivalently here  $B|a, q$ ). All we need in our action model is a way of stating the likelihood of ending up in a particular state. This is simple. Suppose we have the situation in example 2, “Pick up the mug.”, and we have a uniform prior over the candidate objects. If we ask “Is it blue?” then, if  $p'_i$  is the prior for object  $o_i$  then a trivial Bayes update gives the posterior:

$$p''_{i|q,a} \propto p_{a|q,i} p'_i \quad (4)$$

Where  $q$  is the question, and  $a$  is the answer. The only thing we need to know in advance is  $p_{a|q,i}$ , the likelihood of the answer to the question  $q$  (“Is it blue?”) given that object  $o_i$  is the correct binding. This is given by the attributes and relations of the objects as given in the scene graph. If an attribute has the value referred to in the question, or the relation asked about holds, the probability of receiving an answer confirming that attribute or relation, is 1. In our simple case of the question “Is it blue?”,  $p_{yes|q,i} = 1$  for any blue object, and  $p_{no|q,i} = 1$  for any non-blue object. In general, however, what we actually need for the action model is to use these probabilities to estimate the likelihood of making the transition to the posterior belief distribution given by answer  $a$  to question  $q$ , conditioned over the possible objects. For this we use  $p_{a|q,B'}$ :

$$p_{a|q,B'} = \sum_{i \in D} p_{a|q,i} p'_i \quad (5)$$

Which is simply the likelihood given the distribution  $B'$  that the answer to  $q$  will be  $a$ . This is why we are generating the action model on the fly, because  $p_{a|q}$  will depend on the situation ( $B$ ). We also have to normalise over all the possible hypotheses (objects). So if we assume a uniform prior over the distractor set, ask “Is it blue?” and get back the answer “Yes” the posterior will be:

$$B(d_1)|blue?, yes = \{(o2, \frac{1}{3}), (o3, \frac{1}{3}), (o4, \frac{1}{3})\} \quad (6)$$

Note that by taking this approach we are making the assumption that the only possible answers to our questions are

ones that are relevant and do not take into account any conversational implicature of the question asked. We will later incorporate the effects of a human taking into account the conversational implicature. We can therefore build a simple probability tree giving the transitions between the belief distributions given the questions I can ask. Let’s assume the questions are:  $q_1 = “Is it on the left?”$ ,  $q_2 = “Is it blue?”$ ,  $q_3 = “What colour is it?”$ :

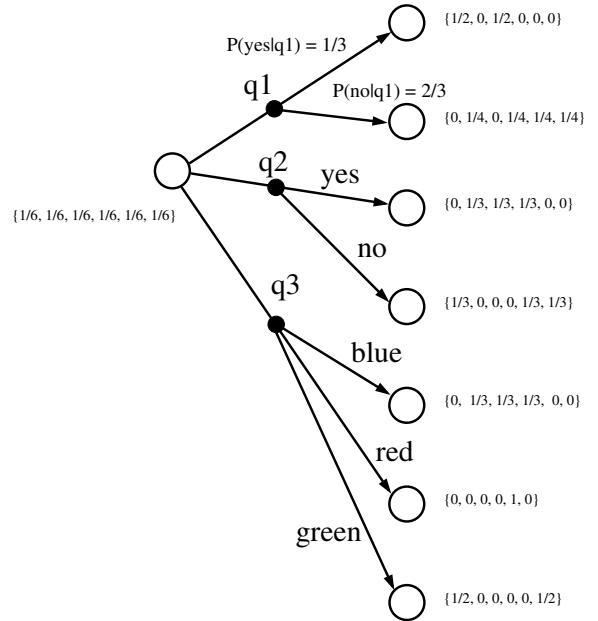


Figure 3: One step action models for three actions.

We may want to consider how to model the effects of multimodal actions, e.g. pointing at an object while speaking about it. We will return to this question later. For now, we look at how we decide that, given the action models, we prefer one dialogue move action over another.

## 5 Evaluating belief distributions

There are a number of criteria we intuitively want to include in our evaluation of a dialogue move and its effects. First we simply want to evaluate the likely degree of ambiguity that remains, clearly our main priority should be to remove the ambiguity altogether if possible. Second, we want a model that will take into account the costs for the other speaker of understanding our clarifying action, and also the likely cost to ourselves of understanding their response.

With respect to the first there are two simple ways we can evaluate distributions over belief distributions. First we can use the entropy of the distribution, and second we can use the expected error rate for the Bayes classifier according to the distribution. In the second we are essentially imagining what our likelihood would be of getting it wrong if we were forced to pick an object. If we have a general distribution then we can calculate the two measures as follows:

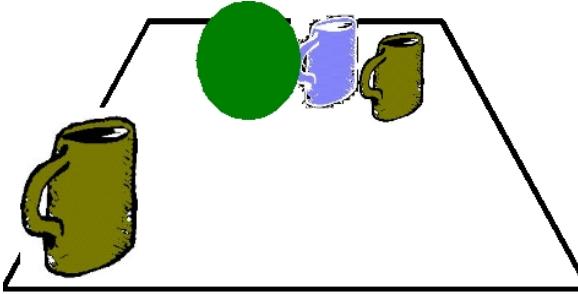


Figure 4: A simple scene. The mug on the left is green, the mug next to the ball is blue, the mug to the right of that is green.

$$Pr(error|q) = \sum_{a \in A} p_{a|q,B} (1 - \max_{k \in B|a} \{p''_{k|a,q}\}) \quad (7)$$

$$E[Entropy|q] = \sum_{a \in A} p_{a|q,B} \sum_{i \in B|a} \{p''_{i|a,q} \log_2 p''_{i|a,q}\} \quad (8)$$

Note that both of these actually calculate the expected value of the belief distributions that could result from a question  $q$ , each of which is weighted by the probability of giving the answer  $a$  that induces the new state. The inner part is the value of the new belief distribution  $B|a$  itself. If we use the expected error rate we obtain very little distinction between most questions. Indeed if we start with a uniform prior as above we can quickly show that all questions with  $N = |A|$  possible answers (where the answers are always relevant) will be equally ranked, and that questions are more preferable the larger  $N$  is. The error rate is rather insensitive to the type of question asked even if the prior over the distractor set is non-uniform. It transpires that the entropy makes it rather easier to distinguish between the effects of questions in reducing the ambiguity. Of course, we would expect an effective question to reduce the ambiguity, and hence the entropy to near zero.

To take into account the cost of understanding the utterances in the planned dialogue we can use the costs that are typically used in the algorithms for generating referring expressions. Imagine the scene in Figure 4, in which the robot has been given the instruction ‘‘Pick up the mug’’. The complete scene graph is given in Figure 5.

It can be clearly seen that we can attach costs to the arcs in the scene graph, and it is precisely these that are used to calculate the ranked costs of possible referring expressions when generating the possible questions. Classically object type is given the lowest cost, followed by absolute attributes, relative attributes, proximal and then projective relations [Dale and Reiter, 1995; Krahmer *et al.*, 2003]. We use these costs, which are generated for the best referring expression for each object, to rate the difficulty to the listener of interpreting the robot’s question. We refer to this as a cognitive load model. The full table of loads is:

The total load imposed by an utterance is simply the sum of the loads involved, e.g. “Is it the green mug to the right of the ball?” carries a load

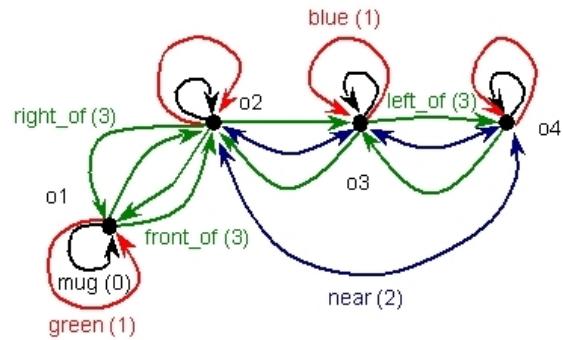


Figure 5: A simple scene graph for the scene in Figure 4. The arcs are labelled with the corresponding attributes and relations, and their associated costs in brackets. Arcs of the same colour are attributes of the same type, and carry the same costs. Not all labels and costs have been included.

Reference by	Example	Load
Type	mug	0
Absolute attribute	red	1
Proximal relation	near	2
Projective relation	left of	3

of  $3 + 1 = 4$ . We assume that questions that make no references, e.g. “Which mug is it?” carry a load of zero. This load is the cost  $C_1$  in Tables 1–4.

Finally we estimate the expected cost to the robot of processing the response. This depends on whether we model the listener’s ability to take account of the implication that the robot has not understood the reference, and is seeking an answer to resolve the reference. We refer to the two types of speaker as helpful and unhelpful. Costs, or loads,  $C_2$  are the expected costs of interpreting the relevant response.

Table 1: Multi-objective evaluation of three questions for a simple scene. We assume an unhelpful speaker, and a prior of 0.5, 0.25, 0.25.

Question	Evaluation		
	$E[H q,a]$	$C_1$	$C_2$
Is it the mug to the left of the ball?	0.5	3	0
Which mug is it?	0	0	2.5
Is it green?	0.69	1	0

Table 2: Multi-objective evaluation of three questions for the same scene, but with a prior of 0.8, 0.1, 0.1.

Question	Evaluation		
	$E[H q,a]$	$C_1$	$C_2$
Is it the mug to the left of the ball?	0.2	3	0
Which mug is it?	0	0	2.8
Is it green?	0.45	1	0

The resulting cost estimates for the case of the scene in Figure 4 above are shown in Table 1. In this case we start with

Table 3: Evaluation of the same questions where we assume a helpful speaker, and a prior of 0.5, 0.25, 0.25.

Question	Evaluation		
	$E[H q, a]$	$C_1$	$C_2$
Is it the mug to the left of the ball?	0	3	1
Which mug is it?	0	0	2.5
Is it green?	0.69	1	.25

Table 4: Evaluation of the same questions, with a helpful speaker and a prior of 0.8, 0.1, 0.1.

Question	Evaluation		
	$E[H q, a]$	$C_1$	$C_2$
Is it the mug to the left of the ball?	0	3	0.4
Which mug is it?	0	0	2.8
Is it green?	0.45	1	0.1

a prior distribution over the distractor set of 0.5 for the mug on the left, and 0.25 for the other two mugs. We assume a non helpful speaker. In this case the speaker will provide a literal answer to a question, without giving additional information on the basis of inferring that the robot wishes to know the object in question. In other words the human doesn't give helpful answers, like "No, it's the blue one." to the question "Is it green?".

For the question  $q_1$ : "Is it the mug to the left of the ball?" we need to generate the possible answers, and estimate their different costs. In summary these costs are: the uncertainty remaining after the question and the answer, the load on the human listener of interpreting the question, and the load on the robot of interpreting the answer. Note that we will not calculate an overall combined cost, but merely rank the costs from most to least important. The entropy given a question is the sum of the likelihoods of each answer by the entropy remaining after that answer has been made, as given in Equation 8.

We can see the likelihoods of the answers, and the resulting belief distributions after the answers in Figure 6. If the answer to  $q_1$  is "Yes", then the posterior belief distribution is  $(1, 0, 0)$  the entropy of which is 0. If the answer is "No" the belief distribution is  $(0, 1/2, 1/2)$ , the entropy of which is 1. So the expected entropy for  $q_1$  is  $H|q = 0.5 \times 0 + 0.5 \times 1 = 0.5$ . We can calculate the expected entropy for the other questions in the same way.

We now need to calculate the expected load of the answer. If the speaker is unhelpful most of the answers are a simple "Yes", or "No" and they carry a cost of 0. In this case only answers to a question like  $q_2$ : "Which mug is it?" will include a referring expression, and thus carry a load. We simply imagine the referring expressions the robot would choose if it were the speaker, and weight their associated costs by the prior. If the answers are: "It's the mug on the left of the ball.", "It's the mug on the right of the ball.", and "It's the blue mug.", the expected load is  $C_2 = 0.5 \times 3 + 0.25 \times 3 + 0.25 \times 1 = 2.5$ .

Of course, if the speaker is helpful, things will be differ-

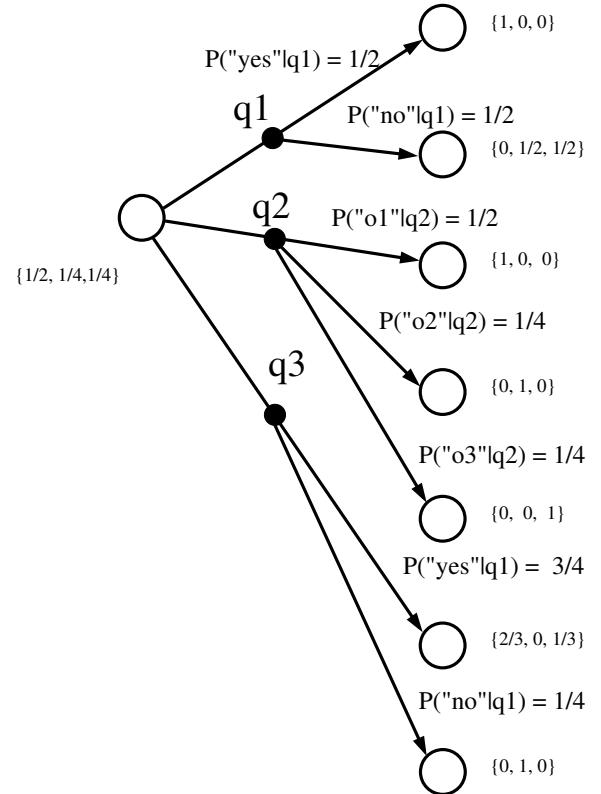


Figure 6: One step model of the three questions for the scene in Figure 4 and the prior  $(0.5, 0.25, 0.25)$ . Note that here, an answer of "o1" is simply a shorthand for a referring expression to object  $o_1$ .

ent, as shown in Table 3. Under this assumption we can assume that the entropy will usually be zero, since if from our clarifying question the speaker can infer our intent they will make an appropriate referring expression. This causes the expected entropy to be 0 for all questions and outcomes except that of answering 'Yes' to the question 'Is it green?'. We assume here that the human would not give additional information since the intention of the speaker is marginally more ambiguous. If the answer always involves a referring expression however, such as that generated by a helpful speaker, then the expected load of processing the answer rises. This would give  $q_1$  a new expected load for the replies to  $q_1$  of  $C_2 = 0.5 \times 0 + 0.25 \times 3 + 0.25 \times 1 = 1$ .

Finally we can see from Table 1 that different orderings of the different types of costs will induce different preferences among the questions. In the unhelpful speaker model, if we wish first to reduce entropy below some threshold (say 0.3), then reduce the cognitive load on ourselves, and then reduce the load on the other speaker we will prefer  $q_2$ : "Which mug is it?". If, however, the prior belief distribution changes enough so that we are more confident that the object is  $o_1$  then we will ask "Is it the mug to the left of the ball?". This is shown in Table 2, where the costs are calculated for the same problem, but with a prior belief distribution of  $(0.8, 0.1, 0.1)$ . The general point here is that

the prior becomes more certain about a specific object it becomes more appealing to ask the question based on the reference to that likely object. This depends on the cost of that reference. Here the cost of a reference to the likely object is high, so that the total costs always exceed those for asking the general question ‘‘Which mug is it?’’.

In the case of a helpful speaker (Tables 3 and 4) we have to decide whether we prefer to lower the load on the robot or the person. An argument for lowering the load on the robot (i.e. preferring questions with a low  $C_2$  to those with a low  $C_1$ ) is that the time taken for the robot to process an utterance by reference to the visual scene will be significantly higher than for the human. Thus to improve speed of response we will typically prefer the specific question involving a referring expression. It is also worth noting that the model will never prefer attribute based questions, such as “Is it blue?”.

Despite this argument it is not entirely clear whether for smooth robot-human dialogue a rational robot should always prefer the higher costs to be placed on a human because of their greater cognitive ability, or whether some other ranking or weighting of the objectives will produce the right mix of behaviour. The flexibility of the cost model is considerable however, so it should not be too challenging to tune it for a reasonable qualitative match with human performance.

## 6 Discussion and Work in Progress

There is a long history of work on question answering systems, many of which are concerned with similar issues. In [Fleming and Cohen, 2001] a related cost model for reasoning about the benefits of different dialogue strategies, including clarification dialogues is proposed. Clarification question planning is also carried out in [Raskutti and Zukerman, 1997], although there the types of uncertainty involve more structured entities, such as more complex discourse relations. There are also similarities with recent probabilistic approaches to dialogue management [Goddeau and Pineau, 2000; Roy *et al.*, 2000]. In those approaches the problem is also posed as either a Markov decision process (MDP), or a partially observable MDP. Our approach should be viewed as myopic planning over an information state MDP, which is, for our purposes, a simplified way of looking at a POMDP. Here the true underlying state of the system is the actual object to which the human has referred. We also remove all learning from the problem, while retaining a more complex (multi-objective) cost function. Finally there are also some connections to the work on incremental production of references to objects. There is strong empirical evidence [Pechmann, 1989] that humans, particularly adults, tend to overspecify references. Pechmann argues that this is due to the incremental nature of speech production. Although referring expressions could be generated incrementally in our system, it does not attempt to be cognitively plausible in that we plan and take into account the precise nature of the references before we have made them. In this sense, our model is not cognitively plausible, although there are opportunities to explore the real impact of visual processing and visual attention in our robot system. Finally we note that because of the challenges of visual processing to obtain object location the sens-

ing problems addressed here are rather different from those of the “Put-that-there” systems of the 1970s [Bolt, 1980]. Indeed the problem of integrating spatial reasoning with language and vision in robotics in a scalable way for natural scenes is still well beyond the state of the art.

We currently have an almost complete communication system that is capable of parsing utterances about objects in a scene, as well as vision routines able to determine the pose of simple objects on a table top, and determine their visual salience. We are currently implementing the planning algorithm described in this paper. There are important issues that crop up when placing such a system in a robot. The most prominent of these is the need for an attentional system that is capable of deciding which parts of the image and scene to process in order to generate a partial scene graph. Generation of a complete scene graph is not feasible for either humans or robots for complex everyday scenes. This will therefore require a mechanism for deciding what processing to do. This is an extremely interesting question because we believe that it might provide a much more satisfactory model of the cognitive load of different referring expressions than merely ranked costs. In addition there is the opportunity for a robot system to estimate its own costs in terms of the processing times and loads for different operations.

We could ask why we would want to put such effort into generating a variety of responses when a general question will do. Indeed why do we really expect references to be commonly underspecified. The answer is that they may not be once context is taken into account, but a robot may well be incapable of spotting and processing all the relevant cues in time, e.g. pointing gestures, or eye fixations. Conversational robots are likely to need a fall back resolution mechanism that reduces the costs for both speakers by attempting to focus attention whenever possible. We believe that such an approach might both provide an interesting account of human reference resolution, as well as making human-robot dialogue more robust. We plan to present initial results from the robot system at the workshop.

Our plans include incorporating action models for watching and making pointing actions. The filters for these are typically probability distributions expressing the observer’s uncertainty about the location to which the arm is pointing. We anticipate that with an appropriate model of pointing that extension to multi-modal actions should be possible.

## Acknowledgments

The author would like to acknowledge the work of Geert-Jan Kruijff, and John Kelleher for the design and implementation of the communication system on which this work is based. He would also like to thank the anonymous reviewers for their useful suggestions.

## References

- [Bolt, 1980] Richard A. Bolt. “Put that there”: Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on computer graphics and interactive techniques*, pages 262–270, 1980.

[Dale and Reiter, 1995] Robert Dale and Ehud Reiter. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[Fleming and Cohen, 2001] Michael Fleming and Robin Cohen. Dialogue as decision making under uncertainty: The case of mixed-initiative AI systems. In *Proceedings of NAACL-2001 Adaptation in Dialogue Systems Workshop*, 2001.

[Goddeau and Pineau, 2000] D. Goddeau and J. Pineau. Fast reinforcement learning of dialog strategies. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2000.

[Kelleher and van Genabith, 2004] J. Kelleher and J. van Genabith. Visual salience and reference resolution in simulated 3d environments. *Artificial Intelligence Review*, 21(3):253–267, 2004.

[Krahmer *et al.*, 2003] E. Krahmer, E.S. van Erk, and A. Verleg. Graph based generation of referring expressions. *Computational Linguistics*, 29(1), 2003.

[Kruijff and Kelleher, 2005] G.J. Kruijff and J.D. Kelleher. A context-dependent model of proximity and regions. In *to appear in Proceedings of the IJCAI-05*, 2005.

[Oates *et al.*, 2000] Tim Oates, Zachary Eyler-Walker, and Paul R. Cohen. Toward natural language interfaces for robotic agents: Grounding linguistic meaning in sensors. In *In Proceedings of The Fourth International Conference on Autonomous Agents.*, pages 227–228, 2000.

[Pechmann, 1989] Thomas Pechmann. Incremental speech production and referential overspecification. *Linguistics*, 27(1):89:110, 1989.

[Raskutti and Zukerman, 1997] Bhavani Raskutti and Ingrid Zukerman. Generating queries and replies during information seeking interactions. *IJCHS*, 47(6):689–734, 1997.

[Roy *et al.*, 2000] N. Roy, J. Pineau, and S. Thrun. Spoken dialog management using probabilistic reasoning. In *ACL*, 2000.

[Roy *et al.*, 2004] Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. Mental imagery for a conversational robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(3):1374–1383, 2004.

[Sidner and Dzikovska, 2004] C. Sidner and M. Dzikovska. A first experiment in engagement for human-robot interaction in hosting activities. Technical Report 2003-134, Mitsubishi Electric Research Labs, December 2004.

[Sidner *et al.*, 2004] C. Sidner, C. Lee, C. Kidd, and N. Lesh. Exploration in engagement for humans and robots. Technical Report 2004-048, Mitsubishi Electric Research Labs, June 2004.

# Assessing the Impact of Adaptive Generation in the COMIC Multimodal Dialogue System

Mary Ellen Foster and Michael White

Institute for Communicating and Collaborative Systems  
School of Informatics, University of Edinburgh  
2 Buccleuch Place, Edinburgh EH8 9LW United Kingdom  
[{M.E.Foster,Michael.White}@ed.ac.uk](mailto:{M.E.Foster,Michael.White}@ed.ac.uk)

## Abstract

We describe how information from the dialogue history and the user model is incorporated into the output-planning process of the COMIC multimodal dialogue system, and present the results of experiments analysing the impact of both of these factors on the generated descriptions. The results of the experiments confirm and extend previous results by showing that both forms of adaptation make a perceptible difference in the generated speech. They also point to ways in which the system output can be improved to take further advantage of these information sources; in particular, they indicate the importance of concessions to negative preferences to the perceptibility of user-model tailoring.

## 1 Introduction

In this paper, we describe how information from the dialogue history and the user model is used to adapt the output-planning process of the COMIC multimodal dialogue system, and present the results of experiments analysing the impact of both of these factors on the generated descriptions.

COMIC<sup>1</sup> (COversational Multimodal Interaction with Computers) is an EU IST 5th Framework project combining fundamental research on human-human interaction with advanced technology development for multimodal conversational systems. The multimodal dialogue system built as part of the project adds a dialogue interface to a CAD-like application used in sales situations to help clients redesign their bathrooms. The input to the system includes speech, handwriting, and pen gestures; the output combines synthesised speech, an animated talking head, deictic gestures at on-screen objects, and direct control of the underlying application.

There are four main phases in the full COMIC system. First, the user specifies the blueprint of their own bathroom, using a combination of speech input, pen-gesture recognition and handwriting recognition. Next, the user chooses a layout for the sanitary ware in the room. After that, the system guides the user in browsing through a range of tiling options for the bathroom. Finally, the user is given a three-dimensional virtual tour of the finished bathroom.

We will concentrate on the third phase of the interaction, where the system helps the user to explore the space of tiling options. The metaphor for this part of the system is one of *guided browsing*, and the goal is for the user to develop a better understanding of the range of possibilities and to have a clearer idea of the features that they like and dislike. The next step is for the user to work with a human sales consultant to fine-tune the exact tiling solution to use in their new room.

The rest of this paper is arranged as follows. In Section 2, we describe how adaptive output was generated and evaluated in several previous systems, and compare them with the COMIC system. In Section 3, we then give details of how adaptive output is created in COMIC using information from the dialogue history and the user model. Then, in Section 4, we describe experiments designed to assess the individual impact on the generated output of each of these knowledge sources. Finally, in Section 5, we summarise the results of the evaluations, and make recommendations for adaptive generation in future systems of this type.

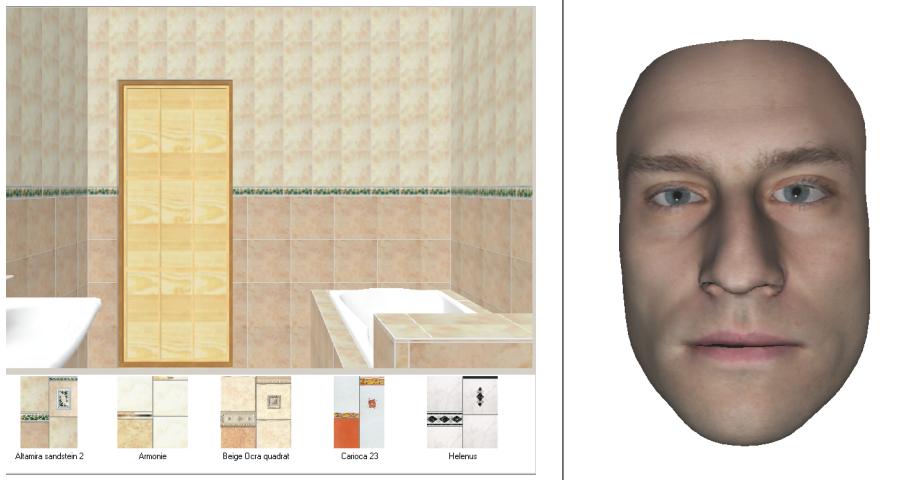
## 2 Evaluating Adaptive Output

The approach we took to adaptive output in COMIC was inspired by the approaches taken in the GEA [Carenini, 2000], M-PIRO [Isard *et al.*, 2003], and MATCH [Walker *et al.*, 2002] systems. In this section, we review the types of adaptation employed in each of these systems, and describe how the impact of the adaptation was assessed. At the end of the section, we compare and contrast the types of adaptation in COMIC with those in the earlier systems, and give an overview of the evaluation reported in this paper.

The GEA system was designed to generate evaluative arguments tailored to user preferences in the domain of real estate, in the context of a graphical data-exploration environment. When generating a description of an option, the user preferences influenced the features that were included, the ordering of those features, and the use of scalar adjectives and adverbs in the text. A task-based evaluation of the generated arguments [Carenini and Moore, 2001] took the following form. While the subjects were using the data-exploration system, a new house was added to the options. A subject was informed about this new house in one of three ways: in some cases, they were just notified that there was a new house, with no description. In other cases, they were given a textual description of the new option; the description was either tailored

---

<sup>1</sup><http://www.hcrc.ed.ac.uk/comic/>



*[Look at screen] THIS DESIGN [circling gesture] uses tiles from SPHINX TILES's HELENUS series. As you can see, there are GEOMETRIC SHAPES on the DECORATIVE tiles. It is in the MODERN style."*

Figure 1: COMIC interface and sample output

to their preferences, or not tailored at all. The effectiveness of the three presentation strategies was assessed by examining the user's behaviour after the presentation of the new option. The authors found that tailored descriptions were significantly more effective than both non-tailored descriptions and no descriptions, while non-tailored descriptions were not any better than no descriptions.

The M-PIRO system generates dynamic descriptions of museum artifacts, with the goal of instructing the user. In the evaluation described by [Karasimos and Isard, 2004], subjects read a series of textual descriptions of artifacts that were generated either with or without sophisticated text structuring. There were two forms of text structuring, which were either both enabled or both disabled together: combining semantically-related propositions into complex sentences (aggregation), and using common attributes to make links between the current object and any previously-seen objects (comparison). In this study, the subjects performed significantly better on a factual recall task when the descriptions they had read included aggregation and comparison; they also gave higher subjective scores to the generated texts that used these structuring techniques.

The MATCH system generates multimodal descriptions and comparisons of restaurants in New York City, tailored to the user's preferences. The user preferences control the restaurants that are recommended to the user, as well as the features of the selected restaurants that are included in the output. [Walker *et al.*, 2004] evaluated the user tailoring by directly asking subjects to judge the quality of the generated output. Subjects were presented with a series of responses to user requests that were tailored either to the subject's own preferences or to the preferences of some other user; the responses were presented in text first and then in speech. They found that subjects significantly preferred descriptions tailored to their own user model, with similar results for both

speech and text.

In COMIC, output is adapted both to the user preferences (as in GEA and MATCH) and to the dialogue history (like M-PIRO). The primary output format in COMIC is synthesised speech, like MATCH and unlike GEA and M-PIRO. The guided-browsing task—in which the system helps the user explore and understand a space with which they are initially unfamiliar—is most similar to the task of M-PIRO; in GEA and MATCH, the role of the system is more one of helping the user to find a relevant option in a space with which they are already familiar. For this paper, we chose to assess directly whether the adaptations were perceptible to users, as in MATCH; we leave for future work questions of whether the adaptation has an impact on task performance and user satisfaction.

### 3 Output Planning in COMIC

Figure 1 shows the interface for the guided-browsing phase of the COMIC system, along with a typical example of the output generated in this phase (where small capitals indicate pitch accents in the speech). The output combines the following modalities:

- Synthesised speech, generated using the OpenCCG surface realiser [White, 2004; 2005a; 2005b] and synthesised using a custom Festival 2 voice [Clark *et al.*, 2004] with support for APML prosodic markup [de Carolis *et al.*, 2004].
- Facial expressions and gaze shifts of the talking head.
- Deictic gestures at objects on the bathroom-application screen, using a simulated mouse pointer.

All of the information about tile designs in the COMIC system is stored in an ontology, which is linked to the dialogue history and the user model. The dialogue manager makes use

of these resources to choose the next set of designs to describe to the user, while the presentation planner uses them to select and structure the content of the descriptions that are generated.

### 3.1 Information and Knowledge Sources

**Ontology** Information about the available tile designs is stored in an ontology represented in DAML+OIL.<sup>2</sup> The ontology contains catalogue information including the manufacturer and series name, style, colours, and decoration, as well as any canned descriptive text associated with each design.

**Dialogue History** For each design in the ontology, the dialogue history keeps track of whether it has been mentioned in the dialogue. It also records the properties of each design that have been described to the user; note that a property may be described directly (e.g., *this design is classic*) or indirectly (e.g., *here are some classic designs* when pointing to several designs). The dialogue history also stores the identity of the last design that was described.

**User Model** As in [Carenini, 2000; Walker *et al.*, 2002; Moore *et al.*, 2004], user preferences are represented in COMIC using a model based on multi-attribute decision theory. Multi-attribute decision models are based on the notion that, if anything is valued, it is valued for multiple reasons. The value of a particular design for a user is computed as the weighted sum of its value on a number of primitive features; to create a specific user’s model, we must therefore set the weights of the attributes and the evaluation function for each individual attribute.

The tile-design user model in COMIC is made up of four features: style, colour, designer, and decoration. These features are represented in a one-level tree. The evaluation function for each of these features assigns a score between 0 and 1 to every possible value of that attribute. A score of 0.5 represents a neutral evaluation; scores above or below that value indicate that the user respectively likes or dislikes that value, with the distance from 0.5 corresponding to the strength of that preference. For the colour and decoration attributes, which may have multiple values on a design, the score is computed by combining the individual attribute values.

The user-model manager supports two types of queries. It can produce an overall evaluation of a set of designs, to help the dialogue manager to choose options that are relevant to the user; it can also produce a detailed evaluation of a single design with scores on each individual attribute, to help the presentation planner create descriptions focussing on the options that are most important for that user. The detailed evaluation of a design is created by retrieving its features from the ontology and computing the evaluation of each feature; the overall evaluation is the weighted sum of the individual evaluations.

There are two ways that a user model can be defined in COMIC. One possibility is that a complete model is created

and stored offline, before the dialogue begins, and is then selected and loaded at the start of the interaction. The other possibility is that the dialogue begins with a neutral user model, which then gets updated during the course of the interaction to take into account emerging preferences. In the current system, this is implemented by increasing the score for attribute values that the user specifically requests. For example, if the user asks for designs with blue tiles, the score for the colour blue is increased in the model for the remainder of the dialogue. There is no support in the current COMIC system for dynamically decreasing scores during the interaction, or for explicitly discussing the user model as part of the dialogue.

### 3.2 Planning Adaptive Output

When browsing through tile designs, the user may request designs with a particular feature—for example, *Show me designs with blue tiles*. To choose the next set of tile designs to present in response to such a user request, the dialogue manager selects all designs from the ontology that meet the user’s criterion, and then uses the dialogue-history and user-model information to rank these designs, favouring those that have not been seen before and those with higher overall scores in the user model.

The dialogue manager then sends a message to the presentation planner that it should describe a particular design to the user. This message contains only a high-level request to describe a particular design, optionally including any features that must be included in the description for dialogue-flow reasons; it is up to the presentation planner to flesh out such a specification into a full description.

The presentation planner selects and structures content from the ontology to meet the dialogue-manager specification, and then creates a logical form for each sentence in the turn. The logical forms are then sent to the OpenCCG surface realiser, which produces prosodically-annotated text that is sent to the Festival speech synthesiser. Festival creates the waveform for the synthetic speech and returns the timing information for the words and phonemes in the text. This timing information from Festival is then used to set the schedule for output in the other modalities (talking-head behaviours, gestures at objects on the screen). When a complete schedule has been prepared for all of the output channels, the presentation planner starts the output. Full details of this process are given in [Foster and White, 2004; Foster, 2005].

The presentation planner has three main decisions to make when creating the content of a design description: choosing the features of the design to include, structuring the selected content appropriately, and choosing the eventual surface form of the resulting text. Information from the dialogue history and user model influences all of these decisions, as follows.

The maximum length of a description is normally three facts about the design being described. The presentation planner always includes any features specifically requested by the dialogue manager; it then chooses the rest of the content (up to the maximum length) based on information from the user model and dialogue history, as follows. First, it includes all features that have not been previously mentioned and that have a non-neutral evaluation in the user model, in decreas-

---

<sup>2</sup><http://www.w3.org/TR/daml+oil-reference>

---

```

<!--
"Once again it is modern, but here the tiles
are from the Carioca collection by Aparici."
-->
<messages>
  <msg type="same-different">
    <slot name="same">
      <msg type="prop-has-val"
          same-as-last="true" prop="has_style">
        <slot name="object" value="Tileset4"/>
        <slot name="value" value="modern"/>
      </msg>
    </slot>
    <slot name="different">
      <msg type="prop-has-val"
          same-as-last="false"
          prop="has_designer-series">
        <slot name="object" value="Tileset4"/>
        <slot name="designer" value="Aparici"/>
        <slot name="series" value="Carioca"/>
      </msg>
    </slot>
  </msg>
</messages>

```

---

Figure 2: Combined text-planning messages

ing priority of user-model score, breaking ties arbitrarily. If there is still space in the description, it then includes any other features that have not previously been mentioned, again in an arbitrary order. Finally, the planner may choose to include a feature with a positive evaluation even if it has previously been described, if it is needed to offset an otherwise entirely negative description. If the result of this process is that no features are selected (because all of the features of the design have already been described), the planner chooses the most highly-ranked previously-mentioned feature to include.

Once the content of a description has been selected, user-model and dialogue-history information is also used to help create an overall structure for the description. When ordering the messages, for example, features with a positive evaluation are generally put earlier in the description. The contextual information is also used to combine adjacent messages into complex sentences where possible. We can combine a feature that is common between the current design and the previous one with one that differs between the two; Figure 2 shows the text planner's internal XML representation of two messages that have been combined in this way. A feature with a high user-model evaluation can be combined with one with a lower evaluation in a similar way.

Finally, the user model and dialogue history also impact the surface form of the description. For example, if a description is to include a fact that we have already told the user, we signal this repetition with words such as *as I said before* or *as I mentioned earlier*. If two facts have been combined to make a comparison with the previous design as in Figure 2, we use a structure such as *Once again X, but here Y*. If we are mentioning a property that we know the user does not like, we add words such as *though* or *although* to the sentence.

## 4 Experiments

We conducted two studies to assess the effectiveness of the adaptive generation described above, looking separately at the impact on the generated output of each of the two knowledge sources; we did not address the dialogue manager's selection of designs to describe. Like [Walker *et al.*, 2004], we used an “overhearer” paradigm in which the subjects watched and listened to recorded interactions between the system and a user and judged the quality of those interactions.

As pointed out by [Whittaker and Walker, 2005], an overhearer evaluation provides several advantages for evaluating multimodal dialogue systems. It allows judgements to be gathered during the course of a dialogue rather than at the end, and allows multiple alternative dialogue strategies to be compared in the same dialogue context. It also avoids any possible problems with speech recognition and language understanding, and allows the evaluation to be run on computers not powerful enough to support the full system. However, it does have the disadvantage that it measures only perception, rather than behaviour change, task performance, or user satisfaction; in Section 5, we discuss how these factors could be assessed.

The interactions were all synthesised in advance, and were presented to the subjects using a modified version of the COMIC system that was able to play back scripted output. User input was provided by playing recordings of a user making requests to the system. This allowed us to ensure that every subject saw and heard exactly the same version of each system turn. The output modules in these experiments were the speech synthesiser and an output-only emulator of the bathroom-design application able to display tile designs and animate pointer gestures; the talking head and full bathroom application were not used.

In both experiments, the basic structure was as follows. Subjects saw and heard two possible versions of a short interaction between a user and the COMIC system. Subjects were then asked to choose which version was better, based on seeing and hearing the presentation. They were then shown the transcripts of both versions of the interaction (in a format similar to Figure 3), and were asked again to choose between the versions. Even though written text is not an output modality of COMIC, we chose to present the transcripts in order to assess whether there were aspects of the adaptation that would have had an effect, but that were missed due to intelligibility issues with the synthesised speech.

Note that our experiment design differs somewhat from that used by [Walker *et al.*, 2004]: the MATCH subjects judged all of the outputs first using text, and then judged them all again in speech, and answered Likert-scale questions to assess the quality of each output individually. We presented the speech first because speech is the primary modality for the COMIC system; the text presentation was shown afterwards to allow the modalities to be compared. We chose to do a forced-choice study instead of using Likert scales because, for a transient medium like speech, direct judgements are difficult to make consistently, especially when the differences are subtle (cf. [Baker *et al.*, 2004; Rocha, 2004]).

- 
- (1) a. **User:** Tell me about this design [*click on Lollipop*]  
     b. **COMIC:** Here is a family design. Its tiles are from the Lollipop collection by Agrob Buchtal. The colours are blue and green.  
     c. **User:** What about this one [*click on Asterix*]  
     d. **COMIC:** Here we have another design in the family style. Its tiles are from the Asterix collection by Villeroy and Boch. It has comic motifs on the decorative tiles.
- (2) a. **User:** Tell me about this design [*click on Lollipop*]  
     b. **COMIC:** This design is in the family style. The tiles are from the Lollipop collection by Agrob Buchtal. The tiles have a blue and green colour scheme.  
     c. **User:** What about this one [*click on Asterix*]  
     d. **COMIC:** This design is in the family style. Its tiles are from the Asterix collection by Villeroy and Boch. As you can see, there are comic motifs on the decorative tiles.
- 

Figure 3: **Also**-class dialogue with dialogue history enabled (1) and disabled (2)

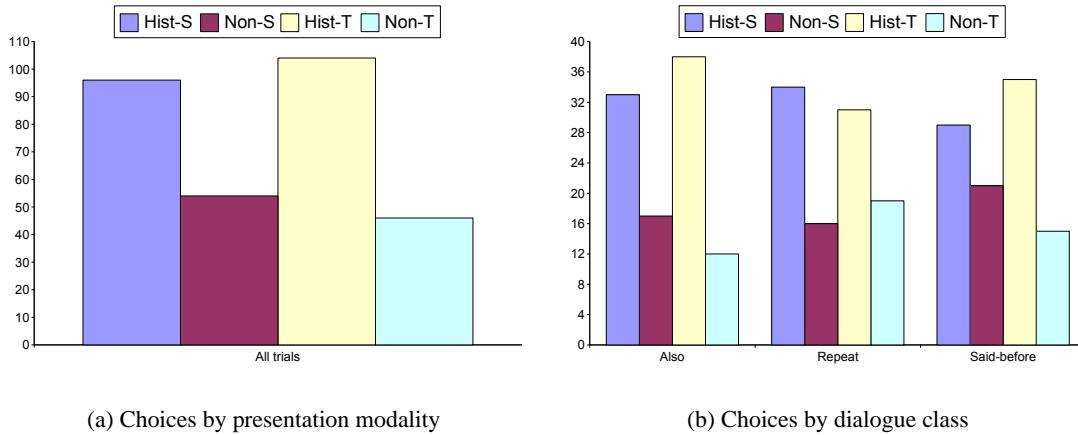


Figure 4: Graphs of dialogue-history results

The two experiments were run consecutively in the same session. Subjects were recruited through an email to the Informatics departmental student mailing list, and were compensated for their participation in the experiment. There were 25 subjects (20 male, 5 female) for the dialogue-history evaluation; due to technical difficulties, only 23 of those subjects were able to complete the user-model study as well. The full study took approximately half an hour to complete.

#### 4.1 Dialogue-History Study

**Materials and Presentation** We created a set of six short dialogues between COMIC and the user, using a neutral user model. Each dialogue had two system turns. The tile designs used in each dialogue were selected arbitrarily, except in those cases where it was necessary that the designs selected have some factor in common. We generated two versions of each dialogue: one in which the second system turn made use of the context provided by the first turn, and one in which it did not.

The set consists of two dialogues in each of the following three classes, representing the three different ways in which the dialogue history can affect the generated output:

**Also** COMIC makes links between features in the current de-

scription and those in the preceding description.

**Repeat** Whenever possible, COMIC avoids repeating information that it has already said about a design.

**Said-before** When COMIC does repeat previously-mentioned information—for example, because there is nothing new to say about a design—it signals the repetition.

Figure 3 shows both versions of one of the dialogues that was generated in class **Also**: (1) makes use of the dialogue history, while (2) does not. The primary dialogue-history-based difference between the two versions is highlighted; all of the other surface differences occur because they were independently generated by the full COMIC presentation planner, which incorporates variation into its planning process.

All 25 subjects in this experiment were shown the same six dialogues, each in an individually randomised order; the order of versions within each dialogue was counterbalanced randomly so that they saw the dialogue-history version first in three trials, and second in the other three. Subjects were instructed to pay attention to how the system responded to the user’s requests and how it kept track of what had already been said in the conversation. After they had seen both versions

- 
- (3) (*target*) Here is a family design. As you can see, the tiles have a blue and green colour scheme. It has floral motifs and artwork on the decorative tiles.
- (4) (*other*) Here is a family design. Its tiles are from the Lollipop collection by Agrob Buchtal. Although the tiles have a blue colour scheme, it does also feature green.
- (5) (*neutral*) This design is in the family style. It uses tiles from Agrob Buchtal's Lollipop series. There are artwork and floral motifs on the decorative tiles.
- 

Figure 5: Output generated for three different user models

of each complete dialogue, subjects were asked the following question: *Which conversation had a more natural sequence of turns?* They chose initially based on the speech presentation, and then chose again after reading the transcripts.

**Results and Discussion** In general, the subjects chose the versions generated with the dialogue history enabled more often than those with it disabled, in both text and speech, as shown in Figure 4(a). Both differences are significant at the  $p < 0.001$  level in a binomial test. Figure 4(b) divides the counts by dialogue class. To analyse statistical significance in this post-hoc analysis of the divided counts, we modified the required  $p$  value using a Bonferroni correction for multiple comparisons; the required significance value to achieve  $p < 0.05$  overall is  $p < 0.017$  on each individual test. All of the differences were significant at this level except for **Said-before** in speech and **Repeat** in text ( $p \simeq 0.16$  and 0.06, respectively).

There was little overall difference between the choices made across the modalities; that they were not equal is probably due to intelligibility issues with the speech. For example, on the **Also**-class dialogue shown in Figure 3, preferences were essentially at chance when subjects chose on the basis of the speech (12–13;  $p \simeq 0.65$ ). However, when subjects were able to read the text, there was a trend<sup>3</sup> for the dialogue-history version (18–7;  $p \simeq 0.02$ ). As highlighted in the figure, the primary difference between the two versions is the single use of the word *another*; this was evidently difficult to pick up based on the speech, but was noticed often when subjects were able to read the transcripts. Adaptations with more obvious surface impacts—e.g., saying *as I said before* to signal a repetition, or describing multiple common properties of two designs—were perceived at a similar rate in text and speech.

## 4.2 User-Model Study

**Materials and Presentation** For this part of the experiment, four random user models were generated. Each model was generated by selecting three feature values that the user liked—which were given an evaluation of 0.8—and five values that the user disliked—which were given an evaluation of 0.2. All other values were given the default evaluation of 0.5. Note that the feature weights were equal in all four user models. One of the random models is shown in Figure 6.

For each model, an individual dialogue with the system was then created. Each dialogue started with the system’s de-

Feature	Likes	Dislikes
Colour	blue, beige	pink
Style		modern, classic
Decoration	floral motifs	geometric shapes
Designer		Porcelaingres

Figure 6: Sample user model

fault selection of designs, and consisted of eight user requests and system responses. The user requests were selected to be plausible for the target user model, and the dialogue-manager also made its choices based on that target model. Four additional versions were then generated of each system output in each dialogue: one version based on the preferences of each of the other three random models, as well as a version based on a neutral user model (all evaluations 0.5). Figure 5 shows three versions of a system turn: (3) is generated based on the user model in Figure 6, (4) reflects the preferences of one of the other user models, while (5) is based on a neutral model.

Subjects were assigned to one of the four target user models in rotation. The target model was shown on screen throughout the study, in a window similar to Figure 6. Subjects were asked to read through the user preferences before beginning, and to keep them in mind when making their choices. Two versions of each system turn in the dialogue were played—the version generated for the target model, and one of the other versions. The target model version was compared with versions for each of the other models twice, in an individually randomly-chosen order; the order within the trials was counterbalanced so that the target model version was seen first in four trials and second in the other four.

After seeing and hearing both versions of each system turn, a subject was asked the following question: *Which COMIC output was more appropriate for this user?* As in the dialogue-history study, subjects chose first on the basis of the speech presentation, and then again after reading the text.

**Results and Discussion** As shown in Figure 7(a), the overall results were similar to those in the dialogue-history evaluation: subjects generally chose the presentation generated for the target model over the one generated for the other model, using both speech and text. These preferences were both significant at  $p < 0.001$ .

In the MATCH evaluation [Walker *et al.*, 2004], the trials were subdivided based on the distance between the target model and the other model. However, the distance measure used there was based on the difference between the feature

<sup>3</sup>The necessary Bonferroni adjusted value for  $p < 0.05$  overall significance is  $p < 0.0083$  on each of the six individual instances.

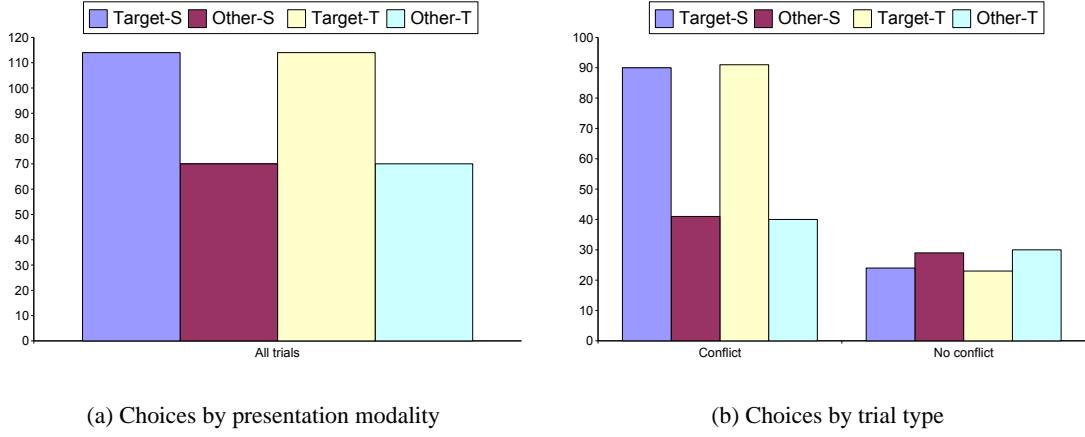


Figure 7: Graphs of user-model results

weights, and is therefore not applicable to the current study in which the weights were the same in all the models. Instead, we divided the trials based on conflicting use of explicit concessions to negative preferences in the two descriptions. For example, the highlighted sentence in (4) has the concession *although the tiles have a blue colour scheme*, as if the user disliked the colour blue; in contrast, (3) has no such concession. This resulted in the following two categories of trials:

**Conflict** There was at least one conflicting concession across the two versions.

**No conflict** There were no concessions at all, or the concessions were the same in both versions.

Note that, due to the way the user models were selected, the majority of trials fell into the **Conflict** class. Figure 7(b) shows the counts for each of these classes. There is a very significant preference for the target description ( $p < 0.00001$ ) in the **Conflict** class, but there is no significant preference either way in the **No conflict** class; the results are similar in both modalities.

The positive preferences do have an effect on the content that is selected for a description and the order in which it is presented: features that the user likes will always be included in the description, and are generally placed nearer the beginning (e.g., compare the content selection in descriptions (3) and (5) in Figure 5). However, this effect was not noticed by the subjects in this study, perhaps because they were basing their judgements on a random user model instead of their own preferences.

## 5 Conclusions and Future Work

These studies together demonstrate that the types of adaptation employed in the COMIC presentation planner do have a perceptible positive effect on the system output. Our results confirm and extend the results of previous studies, and point to ways in which the interaction could be improved to take better advantage of the information stored in the dialogue history and user model.

The results for the dialogue-history tailoring show that it does have a perceptible effect in all cases. This both confirms and refines the results of the M-PIRO study, which presented the output in text and did not examine the effects of aggregation and comparison independently. We also found that with spoken output, dialogue-history adaptations that affect only the surface form may occasionally be missed if the synthetic speech is difficult to understand. However, when the text can be read, such adaptations are perceptible, so with improved speech synthesis they should also have a perceptible effect.

When the output is tailored according to a user model, subjects generally chose the target versions, using both speech and text; this is in line with the results of the GEA and MATCH evaluations. The results of this study demonstrate that the effect of the user-tailored generation in COMIC is only noticeable when there are conflicting concessions to negative preferences in the two versions. Such concessions are also included in the output of GEA, but not in that of MATCH. In general, our subjects could not tell the difference between a description intended to be positive and one intended to be neutral; this may be partly due to the fact that the subjects in our study were basing their judgements on a randomly-generated user model, rather than on their own preferences as in GEA and MATCH. Note that the positive preferences are also used by the dialogue manager to select designs that are likely to be relevant to the user; the effect of this selection was not studied here, but the results of the MATCH evaluation suggest that it would also be perceptible.

While these studies demonstrate that users are able to notice the adaptive generation in COMIC, we have not yet shown that the adaptation has any effect on interactions with the full COMIC system. In future work, we hope to examine this question, using measures such as subjective user-satisfaction scores, objective measures of dialogue quality, and scores on a factual recall task, as in [White *et al.*, 2005]. COMIC could support such an evaluation of the dialogue-history adaptations in its current form; however, a full evaluation of the user-model adaptations would require some modifications to the COMIC system.

To get a true picture of the impact of the user tailoring, the system would have to respond based on the user's actual preferences. In GEA and MATCH, the user model is obtained before the interaction begins, by filling in a form. Note that the domains of these systems—real estate and restaurants, respectively—are ones where users can be expected to have a clear idea of their likes and dislikes before using the system. In contrast, with COMIC, it is not realistic to gather a full user model before the interaction begins, as the purpose of guided browsing is to help users become familiar with the range of available options. Thus, the most natural way to gather preferences would be through the course of the interaction. In the current system, the model is optionally updated by increasing the score for explicitly-requested features; however, there is no way to query or modify the user model that is updated in this way, and the system does not learn any negative preferences.

The system could take more initiative in inferring a user's preferences based on their browsing behaviour, possibly using the work of [Carberry *et al.*, 1999] as a basis. For example, if the user has rejected a series of proposed tile designs that all have green tiles, the system could hypothesise that the user does not like the colour green, and update the model accordingly. Instead of directly updating the user model based on these hypothesis, with no feedback, the system could also ask a confirmation question such as *It appears that you don't like green much*. This would give the user the opportunity to confirm or reject the system's beliefs about their preferences, and should help avoid getting “painted into a corner” prematurely. In the future, we hope to explore extended user-modelling and interaction capabilities in this way, and to assess the effect of the adaptively-generated output on user interactions with the system.

## Acknowledgements

This work was supported by the COMIC project (IST-2001-32311). We thank Johanna Moore, Jon Oberlander, John Lee, Andrea Setzer, Roberta Catizone, and the other members of COMIC for helpful discussions, and the anonymous reviewers for their useful comments on the first draft of this paper.

## References

- [Baker *et al.*, 2004] R. Baker, R.A.J. Clark, and M. White. Synthesizing contextually appropriate intonation in limited domains. In *Proceedings of 5th ISCA workshop on speech synthesis*, 2004.
- [Carberry *et al.*, 1999] S. Carberry, J. Chu-Carroll, and S. Elzer. Constructing and utilizing a model of user preferences in collaborative consultation dialogues. *Computational Intelligence*, 15:185–217, August 1999.
- [Carenini and Moore, 2001] G. Carenini and J. Moore. An empirical study of the influence of user tailoring on evaluative argument effectiveness. In *Proceedings of IJCAI 2001*, 2001.
- [Carenini, 2000] G. Carenini. *Generating and Evaluating Evaluative Arguments*. PhD thesis, Intelligent Systems Program, University of Pittsburgh, 2000.
- [de Carolis *et al.*, 2004] B. de Carolis, C. Pelachaud, I. Poggi, and M. Steedman. APML, a mark-up language for believable behaviour generation. In H. Prendinger, editor, *Life-like Characters, Tools, Affective Functions and Applications*, pages 65–85. Springer, 2004.
- [Clark *et al.*, 2004] R.A.J. Clark, K. Richmond, and S. King. Festival 2 – build your own general purpose unit selection speech synthesiser. In *Proceedings of 5th ISCA workshop on speech synthesis*, 2004.
- [Foster and White, 2004] M.E. Foster and M. White. Techniques for text planning with XSLT. In *Proceedings of NLPXML 2004*, 2004.
- [Foster, 2005] M.E. Foster. Interleaved planning and output in the COMIC fission module. In *Proceedings of ACL 2005 Workshop on Software*, 2005.
- [Isard *et al.*, 2003] A. Isard, J. Oberlander, I. Androtsopoulos, and C. Matheson. Speaking the users' languages. *IEEE Intelligent Systems*, 18(1):40–45, 2003.
- [Karasimos and Isard, 2004] A. Karasimos and A. Isard. Multi-lingual evaluation of a natural language generation system. In *Proceedings of LREC 2004*, 2004.
- [Moore *et al.*, 2004] J. Moore, M.E. Foster, O. Lemon, and M. White. Generating tailored, comparative descriptions in spoken dialogue. In *Proceedings of FLAIRS 2004*, 2004.
- [Rocha, 2004] N.F. Rocha. Evaluating automatic assignment of prosodic structure for speech synthesis. Master's thesis, Department of Theoretical and Applied Linguistics, University of Edinburgh, 2004.
- [Walker *et al.*, 2002] M.A. Walker, S. Whittaker, A. Stent, P. Maloor, J.D. Moore, M. Johnston, and G. Vasireddy. Speech-plans: Generating evaluative responses in spoken dialogue. In *Proceedings of INLG 2002*, 2002.
- [Walker *et al.*, 2004] M.A. Walker, S.J. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28(5):811–840, September–October 2004.
- [White *et al.*, 2005] M. White, M.E. Foster, J. Oberlander, and A. Brown. Using facial feedback to enhance turn-taking in a multimodal dialogue system. In *Proceedings of HCI International 2005*, 2005. To appear.
- [White, 2004] M. White. Reining in CCG chart realization. In *Proceedings of INLG 2004*, 2004.
- [White, 2005a] M. White. Designing an extensible API for integrating language modeling and realization. In *Proceedings of ACL 2005 Workshop on Software*, 2005.
- [White, 2005b] M. White. Efficient realization of coordinate structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 2005. To appear.
- [Whittaker and Walker, 2005] S. Whittaker and M. Walker. Evaluating dialogue strategies. In W. Minker, D. Bühler, and L. Dybkjær, editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*. Kluwer Academic Publishers, 2005.

# User Evaluation of a Conversational Recommender System

Pontus Wärnestål

Department of Computer Science  
Linköping University, SWEDEN  
ponjo@ida.liu.se

## Abstract

Conversational recommender systems (CRSs) approach user preference acquisition from a conversational point of view, where preferences are captured and put to use in the course of on-going natural language dialogue. The approach is motivated by its aim to make interaction efficient and natural, to acquire preferences from the user in a context when she is motivated to give them, as well as to facilitate exploration of the domain and the development of the user's preferences. A CRS's dialogue strategy to achieve these aspects of the interaction is crucial for its performance and usability. This paper reports on a user satisfaction evaluation of ACORN, which is a CRS in the movie domain. The results of the study indicate a high user satisfaction with the interaction from nine usability aspects, and that ACORN's dialogue strategy is suitable for efficient interaction and user preference modeling, and facilitates domain exploration.

## 1 Introduction

Recommender systems produce personalized recommendations of potentially useful items from a large space of possible options (see [Montaner *et al.*, 2003] for a recent overview). To accomplish this, the system needs a model of the user's preferences in the chosen domain, and a recommendation engine that uses the user preference model to predict domain items.

The design of systems aiming at user-adaptive functionality based on user modeling rests on three steps: (1) initialization, (2) update, and (3) usage of the user model to generate adaptive functionality [Wu, 2002]. The initialization step consists of user preference acquisition in order to construct a correct and sufficient preference model [Burke, 2002]. The update part is a more long-term maintenance of the preference model in order to keep it up-to-date with the user's preferences. It involves keeping track of purchased items, changed ratings of items or attributes, and more. User feedback on recommended items is one common way to update the model. After being initialized, the user preference model is put to use by an engine based on one or a combination of several recommendation algorithms.

There are several types of recommendation algorithms available. The two most commonly used today are *collaborative* and *content-based* filtering. Collaborative (or sociological) filtering [Hanani *et al.*, 2001] depends on a user-item matrix where a population of users' ratings of domain items (e.g. books, movies, web pages, etc.) are clustered into "neighborhoods". Users belonging to the same neighborhood thus have similar preferences, and items are recommended to them on the basis that they are relevant to other users in the same neighborhood. In content-based systems, items are defined by properties (e.g. actors, director and genre for movie items), and recommended based on matching item properties to users' preference model contents.

While collaborative filtering represents a major strategy for recommending items and is capable of performing cross-property recommendations, it is not fool-proof. Sometimes users may want recommendations based not only on previous ratings, but rather on explicitly defined rules. This calls for combining different prediction techniques into hybrid engines [Burke, 2002], as well as employing an interaction technique that allows users to state such preferences in a natural and efficient way. Point-and-click information-providing systems often fall short in this context. For example, the Internet Movie Database<sup>1</sup> (IMDB) search engine provides a *Power Search* mode where users can formulate refined queries using standard GUI widgets. The *Power Search* user interface consists of 17 text fields, 20 choice boxes, 6 check boxes, and requires vertical scrolling before a query can be submitted. Despite this complex interface, users are limited in several ways; e.g. only two genres may be combined and there is very limited support for negations. Quantifications and negations are generally hard to capture in graphical user interfaces, but easy to express in natural language [Cohen, 1992; Androutsopoulos *et al.*, 1995].

A *Conversational Recommender System* (CRS) utilizes natural language dialogue between the user and the system where user preferences are initialized, continuously updated, and put to use in order to calculate and present personalized item recommendations. One underlying motivation for the conversational approach is that it aims to exploit situations where the user has a high motivation to provide preference data. This can be contrasted to the "standard approach",

---

<sup>1</sup><http://us.imdb.com>

where preference acquisition burdens the user at initialization time and results in a delayed benefit for the user [Carenini *et al.*, 2003]. This delayed benefit is known as the new-user cold-start problem and is a crucial problem for recommender system performance.

Conversational interaction also seems promising for personalized recommender systems since on-going dialogue may enhance the robustness of the interaction as progress can be made through negotiation [Zadrozy *et al.*, 2000]. This also ties into an *exploratory* interaction aspect: users can learn about new items and concepts in the domain in an incremental fashion throughout the negotiation dialogue and explore the domain, and as a result, evolve their own preferences within it.

The conversational approach is exemplified with the FINDME assisted browsing framework, which is applicable in domains too complex for users to articulate fully specified queries [Burke *et al.*, 1997]. In such situations, incremental refinement in a dialogue fashion is suitable. When faced with a retrieved item in the FINDME framework, the user may articulate some critique of the item. The critique signals what attributes are important for a specific user, and updates the preference model accordingly. Dialogue context can also be utilized to model preference strength depending on the conversational circumstance in which a preference occurs [Carberry *et al.*, 1999]. A recent approach allows users of the ADAPTIVE PLACE ADVISOR system to derive preferences from on-going spoken interaction by gradually narrowing down choices with partial item descriptions [Thompson *et al.*, 2004]. Users are aided in their understanding of the domain and their preferences by thinking about questions in the dialogue. The ADAPTIVE PLACE ADVISOR utilizes dialogue moves to modify the current query and the user preference model.

Since CRSs rely heavily on dialogue and incremental modification of queries and preference models, a system's *dialogue strategy* to initialize, update, and use the preference model in the interaction is crucial for the system's performance and usability. The work presented here reports on a user satisfaction evaluation of the CRS ACORN<sup>2</sup> that employs a dialogue strategy covering these aspects. An experiment with twenty users that carry out tasks in a pre-given scenario using ACORN has been conducted, and by analyzing the dialogue sessions and users' responses to a questionnaire, nine aspects of user satisfaction are assessed.

## 2 ACORN

ACORN is a text-based dialogue system built specifically to implement and evaluate a recommendation dialogue strategy in the movie domain. Figure 1 shows an excerpt of a dialogue with ACORN, and illustrates some key points of its dialogue strategy:

- **System-driven preference acquisition:** the system's agenda is to interview the user for her movie preferences. User responses are utilized to gather preference data that can be used in future recommendations (e.g. utterances A2 and A4 in Figure 1).

<sup>2</sup>Adaptive COnversational RecommeNder

- **User-driven information requests:** at any time the user is allowed to take initiative and issue information requests, and let the user retrieve desired pieces of information about items in the domain as in utterances U5 and U8. In effect, this drives the dialogue forward, shown in utterances A5-U5-A6-U6 (since Sean Connery appears on the actor list in A6).
- **Recommendations in a dialogue context:** the recommendations are presented and properly motivated in a personalized manner within the on-going dialogue (e.g. A8).

```

A1: How may I be of service?
U1: I feel like watching a comedy
A2: OK. Can you tell me one of your favorite actors or actresses?
U2: I think Brad Pitt is good
A3: I see. Have you seen the movie Snatch (2000)?
U3: Yes
A4: What do you think about that movie?
U4: It is great
A5: Do you want another recommendation?
U5: Who is acting in the movie Entrapment?
A6: Entrapment (1999) has the following cast. (ACORN displays a list of actors: Sean Connery, Catherine Zeta-Jones, ...)
U6: Can you recommend a drama with Sean Connery?
A7: OK. Have you seen Finding Forrester (2000)?
U7: No
A8: Finding Forrester (2000) is a drama starring Sean Connery.
U8: Who else is starring in it?

```

Figure 1: Recommendation dialogue about movies between the ACORN system (A) and a user (U). The dialogue has been translated from Swedish.

The user interface (see Figure 2) consists of a chat-style panel where the dialogue between ACORN and the user takes place and a text field where the user types her input. To the right of the chat panel is a result presentation panel where movie information and other pieces of domain information is displayed.

ACORN's architecture is phase-based [Degerstedt and Johansson, 2003] and builds on the MOLINC toolkit<sup>3</sup>. The main components of ACORN are: a **dialogue manager** (including the recommendation dialogue strategy and a standard information-providing dialogue system), a **domain knowledge manager** (including a basic content-based recommendation engine and a movie database), and a **preference manager**.

### 2.1 Dialogue Management

ACORN's dialogue capabilities are divided into two initiative types: system-driven preference acquisition and recommendations, and user-driven information requests. The dialogue behavior is based on an empirical investigation of human-human recommendation dialogues, and has been adapted to

<sup>3</sup>Available at <http://nlpfarm.sourceforge.net/>.



Figure 2: The ACORN user interface.

human-computer dialogues by the process of dialogue distilling [Larsson *et al.*, 2000]. Dialogue-specific modules include a state graph describing the preference acquisition and recommendation strategy (see Figure 3); a phase graph [Degerstedt and Johansson, 2003] describing information-providing functionality, and that facilitates local focus shifts and anaphora; and standard linguistic resources such as a lexicon and a grammar for the interpreting module.

### Preference Acquisition and Recommendations

The preference acquisition takes place as a series of questions to the user, much like an interview guide. User responses are captured, and the interpretations are used to decide what question to issue next. The strategy consists of sixteen dialogue states depicted in Figure 3. Each state (except for the square nodes) corresponds to a specific preference and recommendation question or statement. The *RecEngine* node is the interface to the back-end recommendation engine which is described below. All ACORN’s utterances in Figure 1 except A6 are examples of individual dialogue states in the model. Transitions between states depend on the interpreted user utterance. A more detailed description of the dialogue strategy and its implementation is found elsewhere [Wärneström, 2005; Johansson, 2004].

### Information Requests

While the preference acquisition and recommendation part of ACORN dialogues are system-driven, users can take initiative and ask for domain information at any time in the dialogue. The turn U5-A6 in Figure 1 are examples of user-initiated information requests.

This information-providing part of the dialogue manager consists of a phase-based component consisting of four phases that operate on a data structure holding information on interpretation, dialogue move, task, and generated output. The phase graph can be accessed from each of the above mentioned preference dialogue states, allowing the user to issue information requests at any time in the dialogue.

The **Linguistic Analysis** phase uses a parser module that produces a task representation of the user utterance. In the **Pragmatic Interpretation** phase a refined interpretation

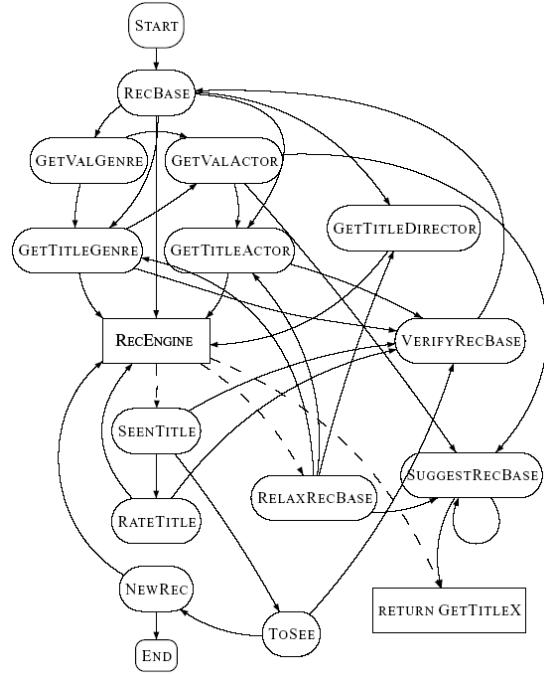


Figure 3: ACORN’s recommendation dialogue strategy. Rounded nodes correspond to system utterances, square nodes to system actions. Solid transitions correspond to interpreted user utterances, and dashed to system logic decisions. Some transitions have been omitted for clarity.

based on dialogue context is carried out by using a dialogue memory to add/change information in the task representation. The Pragmatic Interpretation phase is required for simple sub-dialogue capabilities, such as asking for clarifications or additional information, or refinements if the database returns too many or no hits. The **Task Handling** phase executes the task by retrieving information from the database. The result set is transformed to suitable output in the **Generation** phase with slot-filling templates.

Whenever an information request has been addressed by ACORN, the preference and recommendation dialogue continues to gather preferences and provide recommendations, until a new information request is detected. For more details see [Johansson, 2004].

Responses to information requests are resolved without the preference model and recommendation engine filtering. Furthermore, information requests are not modeled in the preference model. This means that users are free to ask for any kind of movie information without having to worry about how or if their information requests affect future recommendations.

## 2.2 Domain Knowledge and Recommendation Engine

ACORN derives information and recommendations from a local copy of IMDB that covers information about 230,000 movie titles. The searchable features are currently limited to actor, character, director, genre, and plot information for each movie.

The recommendation engine is a basic content-based module that calculates recommendations on actor, genre, and director preferences. A more sophisticated recommendation engine utilizing e.g. collaborative filtering for cross-property recommendations in addition to the current content-based component would be desired [Swearingen and Sinha, 2002], but for the purpose of this study the mechanisms described above are deemed sufficient. A collaborative filtering approach relies on item (in this case movie title) ratings. This is supported by the dialogue strategy and could thus be used with a hybrid recommendation engine without modification. The current engine does keep track of movie ratings from the IMDB data set (February 2005) in order to return the highest ranked movie in the community that matches the current user preference model, but it does not maintain a user-item matrix required for collaborative filtering.

The engine's output after a request from the dialogue strategy can be of three types: (1) the highest ranked movie title based on content-based predictions of the user preference model; (2) a message indicating that the current recommendation base has been exhausted (i.e., there are no more movies matching the user's criteria); or (3) a message indicating that more information is needed in order to perform a recommendation. These messages are shown in Figure 3 as dashed transitions leading from the *RecEngine* node. The *RecEngine* node can thus be seen as a guard before entering the *SeenTitle* node that actually recommends a movie to the user.

For example; in the excerpt shown in Figure 1, the recommendation in A3 is based on a genre and an actor. Since the matching movie *Snatch* (2000) has the highest ranking and has not previously been recommended to the user, it is suggested.

### 2.3 Preference Management

A user's preference model is constructed during the dialogue interaction and consists of instances of genres, actors, directors, and titles that have occurred in the dialogue(s). Users may provide both positive and negative preferences. The positive preferences correspond to the matching set the system tries to retrieve from the database (e.g. comedies starring *Brad Pitt* in Figure 1). Negative preferences (if existing) act as filters to the matching set. Seemingly conflicting preferences such as utterance U2 in Figure 4 thus affect the dialogue flow. The current recommendation base is "comedies", and the only actor preference we have detected so far is *Brad Pitt*. However, all comedies with *Brad Pitt* has gained a low rating, so the recommendation engine signals that the current recommendation base has been *exhausted* (see Figure 4), prompting the transition to the *RelaxRecBase* dialogue state in Figure 3 that asks for more preferences (A5). Future recommendations may thus include movies starring *Brad Pitt*, so long as they are not comedies.

## 3 Experimental Design

The study was designed to test the following nine aspects of user satisfaction with ACORN:

- Task ease: How well the user feel that a particular task can be carried out with the system.

A1: How may I be of service?  
U1: I would like a drama  
A2: OK. Can you tell me one of your favorite actors or actresses?  
U2: I like *Brad Pitt* but I can't stand his comedies  
A3: I see. Have you seen the movie *Fight Club* (1999)?  
U3: No  
A4: *Fight Club* (1999) is a drama starring *Brad Pitt*.  
U4: Can you recommend a comedy?  
A5: OK. Are there any other actor or actress you prefer?

Figure 4: Dialogue excerpt showing preference updates in ACORN. The dialogue has been translated from Swedish.

- Adaptation: How well the system adapts to an individual user's preferences.
- System response time and pace: How fast the system responds, and whether the interaction pace feels satisfactory for the user.
- Domain coverage: Measures whether there are enough items in the domain to solve a task, and whether there is enough information about each item.
- Interpretation performance: The user's experience of how well the system understands her input.
- Generation performance: How well the system performs when generating linguistic responses (phrase choice, clarity, and verbosity).
- Expected behavior: Measures how intuitive and natural the dialogue interaction is, in terms of initiative and grounding etc.
- Entertainment value: Assesses how entertaining and interesting it is to engage in a dialogue with the system.
- Future use: Whether it is likely that the user will use the system in the future or not.

These factors are assessed by analyzing (a) dialogues from the user sessions, and (b) a post-study questionnaire filled out by the each of the participants.

The questions are inspired by the PARADISE dialogue system evaluation framework [Litman and Pan, 1999, page 59]. The PARADISE framework is for evaluating spoken dialogue systems, with a focus on task-oriented systems [Walker *et al.*, 1997]. In this study the set of user satisfaction aspects—and corresponding questions in the questionnaire—were enhanced in order to address e.g. entertainment value and adaptation assessment.

### 3.1 Participants

Twenty participants of varying age, gender, and background were recruited as users. None of them had any special knowledge of dialogue systems, but were all proficient computer users. They were not paid.

### 3.2 Procedure

Each participant received a quick tutorial explaining rudimentary facts about ACORN (e.g. it being a movie information and recommendation system, that the interaction is typed, and that

the language of interaction is Swedish, etc.). Next, the participant was presented with a short scenario consisting of three sub-tasks (see Appendix A). The tasks ranged from strict (tasks 1 and 3) to a bit more open-ended (task 2) in order to ensure that the sessions are comparable (cf. [Walker *et al.*, 1997]), but at the same time allow for some variety in the solutions. In addition to the comparable quality, strict tasks are also a way to determine efficiency (i.e. that a specific task is resolved), since users have no reason to persist in artificial constructed open-ended tasks and may settle for almost anything if there is no “real” or personal motivation. During the dialogue session, the participants noted their solutions/results of each task on a protocol. After completing the scenario, they were asked to fill out the questionnaire. The questionnaire consists of 23 questions regarding user attitudes towards task solution, system performance, and dialogue interaction. Response values are encoded on an ordinal scale of 1–4 corresponding to the statements *I strongly disagree* (1), *I somewhat disagree* (2), *I somewhat agree* (3), and *I strongly agree* (4). The sessions were also logged during the interaction and time-stamped and saved on file.

## 4 Results

The study yielded two kinds of data. First, the dialogue logs constitute data for a dialogue corpus analysis. Second, the questionnaire responses provides data for a user satisfaction analysis.

### 4.1 Dialogue Corpus Analysis

Session logs of the interactions resulted in a corpus with a total of 226 complete turns, and a total elapsed time of 4 hours (mean 12 minutes per dialogue). The corpus was annotated manually with the number of system interpretation failures, and the number of system restarts. Furthermore, each dialogue was compared to an “optimal” scenario solution which represent the shortest number of turns that are required to solve all sub-tasks in the scenario. The scenario can be completed in seven turns, which is the “key” to an efficient dialogue. However, since the scenario can be resolved in a variety of ways, an additional turn or two may still feel both optimal and natural for a particular user. All twenty subjects accomplished all tasks in the scenario. The average number of turns for completing the tasks is 11.3, and 10 subjects accomplished the scenario within the optimal range (7–9 turns, depending on their strategy, and personal choice in the open-ended task).

Dialogues longer than 7–9 turns are the result of (i) system interpretation failures (due to a variety of factors, such as lack of linguistic coverage on the system’s part, or on uncooperative behavior or misspellings on the user’s part, etc.), (ii) domain exploration (e.g. asking for more recommendations, or additional information not required by the scenario), or (iii) miscellaneous turn types, such as clarification sub-dialogues due to too many database hits, etc.

Table 1 shows some interesting aspects of the data, which will be discussed below.

User	TRN	FAL	EXP	MSC	SAT
01	14	6	0	1	2.5
02	7	0	0	0	3.1
03	9	1	1	0	2.7
04	14	0	2	5	3.2
05	9	1	0	1	3.3
06	12	1	0	4	3.2
07	15	0	5	3	3.3
08	17	1	5	4	2.7
09	17	0	6	4	3.2
10	7	0	0	0	3.3
11	7	0	0	0	2.9
12	7	0	0	0	2.9
13	7	0	0	0	3.3
14	8	0	0	1	3.0
15	23	7	1	8	2.6
16	10	0	0	3	2.8
17	9	2	0	0	3.0
18	9	0	1	1	3.0
19	12	2	2	1	2.8
20	13	3	3	0	3.0
Total	226	24	26	36	N/A
Mean	11.3	1.2	1.3	1.8	3.0

Table 1: Total number of dialogue turns (TRN), number of interpretation failure turns (FAL), domain exploring turns (EXP), miscellaneous turns (MSC), and mean user satisfaction over the nine categories from the questionnaire (SAT).

**Interpretation failure turns (FAL)** are defined as turns that were not successfully parsed by the system, i.e. not covered by ACORN’s linguistic resources. The total interpretation failure rate for the complete corpus is 1.2 turns per dialogue, with twelve completely error-free dialogues.

**Domain exploring turns (EXP)** are requests that are “not necessary” in order to complete scenarios. Examples include asking for plot information even though no task in the scenario requires that kind of information, or asking for more recommendations than required. Nine of the users used one or more turns for such additional domain exploring. The dialogue excerpt in Figure 5 is an example of when the user asks for more information and recommendations than required by the scenario.

**Miscellaneous turns (MSC)** include a variety of turns, and have deliberately been merged together for the purpose of this analysis since they are not in focus here. These turns include e.g. sub-dialogue clarifications when the database returns none or too many hits. Another example of turns in this category arises due to the constructed and artificial nature of the scenario: A user may for example not like the actor Adam Sandler, and may dislike his comedies even more; and this has an impact on the dialogue of Task 3 (see Appendix A), since users may respond negatively to the resulting recommendation. This causes ACORN to spend turns trying to find out what is wrong with its preference model of the user. This phe-

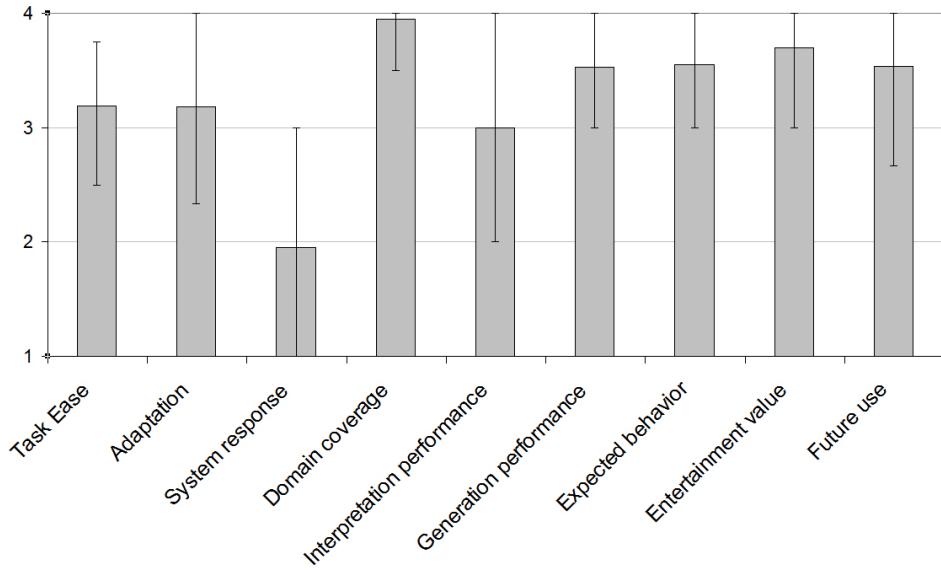


Figure 6: User satisfaction based on the subjects’ own assessment split into nine categories based on the post-study questionnaire. Response values are: *I strongly disagree* (1), *I somewhat disagree* (2), *I somewhat agree* (3), and *I strongly agree* (4).

- U1: please recommend a drama with *Sean Connery*
- A1: Have you seen the movie *Action of the Tiger (1957)*?
- U2: no
- A2: *Action of the Tiger (1957)* is a drama starring *Sean Connery*. Do you want another recommendation?
- U3: yes please
- A3: Have you seen the movie *Anderson Tapes, The (1971)*?
- U4: what is it about?
- A4: Here is the plot for *Anderson Tapes, The (1971)*. Have you seen it?
- U5: no
- A5: *Anderson Tapes, The (1971)* is a drama starring *Sean Connery*. Do you want another recommendation?
- U6: yes a comedy with *Adam Sandler*

Figure 5: Excerpt where the user deviates from the scenario requirements in order to explore the domain. The turns U3-A5 are clearly outside the scope of the scenario since it only requires one recommendation based on the attributes in U1, and no additional information requests such as U4. The dialogue has been translated from Swedish.

nomenon would not arise in “real” situations because users disliking Adam Sandler’s comedies would not ask for such recommendations.

**User satisfaction (SAT)** is a metric that consists of mean values of the questionnaire responses (graded 1–4) for each of the nine aspects. The mean values should naturally be interpreted with care, since the questionnaire provides an ordinal scale. However, they are still useful indicators for the upcoming user satisfaction analysis.

## 4.2 User Satisfaction Analysis

The 23 questionnaire items were grouped into the nine categories, and the responses were weighed together. For example, for the category Adaptation users responded to the following questionnaire:

1. ACORN’s responses were relevant and helped me solve the tasks.
2. ACORN’s recommendations were effective and matched the preferences I had put in.
3. ACORN adapted continuously to my preferences.

Figure 6 shows the results of each of the nine user satisfaction categories for all twenty users.

It is noteworthy that nine out of twenty users engage in domain exploration dialogues. This should be considered rather high, since the users were *not* instructed or even encouraged to engage in such dialogues. Domain Coverage (mean score 3.9) and Entertainment Value (3.7) are the two highest-ranking aspects, and users engaging in exploration turns give the highest entertainment value ratings.

In the higher range we also find Expected Behavior (3.6), Generation Performance (3.5) and Future Use (3.5). Adap-

tation (3.2) is slightly lower, and could be affected by that the given scenario contains tasks that do not fit certain users, such as the given choice of actors and genres in task 3. Another important factor to consider is that adaptation typically requires more long-term use than just one session.

System Response Time (2.0) is significantly lower than the other rankings and is due to the slow response-time because of the large database.

## 5 Discussion

The study shows that ACORN’s dialogue strategy allows for *efficient* dialogues, since all users accomplished the scenario, and that several even completed the tasks within the optimal number of turns. This capability may be seen as a prerequisite for CRSs, not to say for dialogue systems in general.

The low interpretation error rate would suggest that the user satisfaction rates are indeed measuring the desired aspects, without them being influenced by a general dissatisfaction with interpretation performance of the system.

One interesting observation is that the dialogue interaction has an *entertaining* quality. The number of domain exploring initiatives suggests that users finds the interaction interesting enough to deviate from the scenario, and engage in the dialogue out of personal interest. Exploratory behavior seems to happen towards the middle or end of the dialogue sessions, which indicate that such behavior is not only an attempt to familiarize with the system. Users engaging in exploration turns also give the highest Entertainment Value ratings. This suggests that exploring the large domain space is an entertaining feature of interacting with ACORN. The questionnaire analysis shows that this is one of the most high-ranked satisfaction aspects. It is thus clear that dialogues longer than the “optimal efficiency” may still have a high user satisfaction value attached. This is especially true in exploratory dialogues where it is a quality in itself for the user to be able to learn more about the domain than initially planned. Obviously, this comes with the domain and the purpose of the system. Exploring the movie domain and receiving personalized recommendations is different from finding train table information as quickly as possible.

As pointed out in Section 1, previous research suggests that an important reason for investigating and developing CRSs is that they allow the system to capture user preferences when users are as motivated as possible to give them. One promising way to achieve this is to employ a dialogue strategy with entertaining dialogues that encourage domain exploration but still allows for efficient handling of user tasks.

An interesting fact is that the users’ opinion on Interpretation Performance according to the questionnaire seems rather low compared to the factual number of interpretation failures (FAL rates in Table 1). When examining these dialogues in detail it seems that users that encountered an interpretation failure in the *beginning* of their session, did not read ACORN’s responses as carefully in the remainder of the session. An example of this is found in Figure 7 where utterance U1 was not understood by ACORN, resulting in the help message A1. After the additional help message the user does not seem to care much about ACORN’s prompts, and fails to realize that

her request has been addressed correctly. This stresses the

- U1: I want to know about *Brad Pitt*
- A1: To be able to provide recommendations I need to know about your movie preferences. Say for example: “I would like a drama”.
- U2: a movie starring *Brad Pitt*
- A2: Ok. I need more information in order to give you a recommendation. You can for example provide a genre you’re interested in.
- U3: comedy
- A3: Have you seen the movie *Snatch (2000)*?
- U4: I want a comedy starring *Brad Pitt*, can you do that?

Figure 7: Excerpt showing a user’s failure to realize that a correct recommendation has been given. (The recommended movie *Snatch (2000)* is a comedy starring *Brad Pitt*, which is what the user asked for). The dialogue has been translated from Swedish.

importance of interpretation performance since a lack thereof takes away the feeling of interacting with a competent conversational partner, and this causes the user to pay less attention to the system’s utterances and dialogue.

Another possible explanation for this behavior is concerned with *when* to issue explanations. Objective explanations of recommendations serve two key points in the interaction: to provide transparency of the system logic, and inspire trust between the user and system [Swearingen and Sinha, 2002]. In the original human-human corpus—on which ACORN’s dialogue strategy is based (see section 2.1)—utterance sequences such as A7-U7-A8 in Figure 1) are standard. That is, the recommender provides an explanation of the recommendation only if the respondent has *not* seen the recommended title (compare to A3-U3-A4 in Figure 1). However, providing the explanation of the recommendation afterwards seems not to be suitable for the dialogue in Figure 7. Here, it would perhaps be better if the explanation that *Snatch (2000)* is in fact a comedy starring *Brad Pitt* could be provided as part of utterance A3. A future version of ACORN’s dialogue strategy could thus present recommendation explanations earlier; or better yet provide *adaptive* recommendation explanations that depends on e.g. the number of previous successful transactions and recommendations.

## 6 Summary and Future Work

The results of the study indicate a high user satisfaction with the interaction from nine usability aspects, and that ACORN’s dialogue strategy is suitable for both efficient interaction and domain exploration. ACORN not only suggests recommendations but also provides means to explore the domain in an information retrieval dialogue at any point in the conversation. Entertainment value is one of the highest-ranking aspects of the interaction, and it is suggested that this is a suitable interaction quality to make use of in order to facilitate domain exploration and preference modeling for conversational recommender systems.

Future work includes a long-term user study of ACORN, where user preference model updates in multiple dialogue

sessions are tracked and used. Such a study would require a more sophisticated recommendation engine, as well as preference manager that is able to deal with preference changes over larger time spans. In a long-term study it would be important not to have the artificial agenda in the form of a given scenario, but to study naturally occurring domain exploration of the type hinted at in this study.

## Acknowledgments

This research has been supported by the Swedish National Graduate School of Language Technology (GSLT), Santa Anna IT Research, and VINNOVA. I am grateful to my advisor Arne Jönsson for his valuable comments on previous versions of this paper.

## A Abbreviated Scenario Instructions

The scenario presented to the participants consisted of three sub-tasks. They are presented here in an abbreviated form.

### A.1 Task 1

Find out if the actor *Brad Pitt* has acted in any comedies. Please mark your answer in the protocol.

### A.2 Task 2

Find out what actors/actresses are starring in the movie *Entrapment*. Note the name of one of them, and ask ACORN to recommend a drama starring that actor/actress. Please note the actor/actress and recommended title in the protocol.

### A.3 Task 3

Ask for a comedy starring *Adam Sandler*, and note the recommended title in the protocol. Then find out who has directed the movie, and note his/her name in the protocol.

## References

- [Androultsopoulos *et al.*, 1995] I. Androultsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases—an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.
- [Burke *et al.*, 1997] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The FindMe Approach to Assisted Browsing. *IEEE Expert*, 12(4):32–40, 1997.
- [Burke, 2002] Robin D. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.
- [Carberry *et al.*, 1999] S. Carberry, J. Chu-Carroll, and S Elzer. Constructing and utilizing a model of user preferences in collaborative consultation dialogues. *Computational Intelligence*, 15(3):185–217, 1999.
- [Carenini *et al.*, 2003] Giuseppe Carenini, Jocelyn Smith, and David Poole. Towards more conversational and collaborative recommender systems. In *Proceedings of the International Conference of Intelligent User Interfaces*, pages 12–18, Miami, Florida, USA, 2003.
- [Cohen, 1992] Philip R. Cohen. The role of natural language in a multimodal interface. In *Proceedings of the 5th symposium on user interface software and technology*, pages 143–149, 1992.
- [Degerstedt and Johansson, 2003] Lars Degerstedt and Pontus Johansson. Evolutionary Development of Phase-Based Dialogue Systems. In *Proceedings of the 8th Scandinavian Conference on Artificial Intelligence*, pages 59–67, Bergen, Norway, 2003.
- [Hanani *et al.*, 2001] Uri Hanani, Bracha Shapira, and Peretz Shoval. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, 11:203–259, 2001.
- [Johansson, 2004] Pontus Johansson. Design and development of recommender dialogue systems. Licentiate Thesis 1079, Linköping Studies in Science and Technology, Linköping University, April 2004.
- [Larsson *et al.*, 2000] Staffan Larsson, Lena Santamarta, and Arne Jönsson. Using the process of distilling dialogues to understand dialogue systems. In *Proceedings of the 6th International Conference on Spoken Language Processing*, 2000.
- [Litman and Pan, 1999] Diane J. Litman and Shimei Pan. An empirical evaluation of an adaptable spoken dialogue system. In *Proceedings of the Seventh International Conference on User Modeling*, pages 55–64, 1999.
- [Montaner *et al.*, 2003] Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19:285–330, 2003.
- [Swearingen and Sinha, 2002] Kirsten Swearingen and Rashmi Sinha. Interaction design for recommender systems. In *Interactive Systems (DIS2002)*, London, June 2002.
- [Thompson *et al.*, 2004] Cynthia Thompson, Mehmet Göker, and Pat Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.
- [Walker *et al.*, 1997] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proc. 35th Annual Meeting of the Association for Computational Linguistics*, pages 271–280, 1997.
- [Wärnestål, 2005] Pontus Wärnestål. Modeling a dialogue strategy for personalized movie recommendations. In *Proceedings of Beyond Personalization Recommender System Workshop at IUI'05*, pages 77–82, San Diego, CA, USA, January 2005.
- [Wu, 2002] Hongjing Wu. *A Reference Architecture for Adaptive Hypermedia Applications*. PhD thesis, Eindhoven University of Technology, 2002.
- [Zadrożny *et al.*, 2000] W. Zadrożny, M. Budzikowska, J. Chai, N. Kambhatla, S. Levesque, and N. Nicolov. Natural language dialogue for personalized interaction. *Communications of the ACM*, 43(8):116–120, 2000.

# Generating Confirmation to Distinguish Phonologically Confusing Word Pairs in Spoken Dialogue Systems

Kazunori Komatani

Ryoji Hamabe

Tetsuya Ogata

Hiroshi G. Okuno

Graduate School of Informatics

Kyoto University

Yoshida-Hommachi, Sakyo, Kyoto 606-8501, Japan

komatani@i.kyoto-u.ac.jp

## Abstract

The intelligibility of responses in spoken dialogue systems is important for communicating successfully, especially when the systems are used via cellular phones in noisy environments. We contrive language expressions of system responses to avoid users misunderstandings, which may occur when word pairs in a system's vocabulary phonologically resemble each other. We designed and developed a method to automatically add clarifying expressions when needed. Our method can be applied to various domains because no handwritten rules are needed in new domains. It exploits multiple knowledge sources such as dictionaries to get distinctive words as candidates to use in clarifying expressions. To select the best one, we introduce a criterion: ease in hearing. It is defined as the weighted sum of the uniqueness of the distinctive word and the difference between the distinctive and original words. We evaluated our method by applying it to the vocabularies of two systems. An experiment with five subjects showed that our system improved the intelligibility of phonologically similar words.

## 1 Introduction

Because of recent improvements in spoken dialogue systems, we can access information systems verbally with cellular phones [Komatani *et al.*, 2003; Raux and Eskenazi, 2004]. Since no special new apparatuses are needed when the telephone is used as the interface, such information services are expected to be used by various people, including the elderly.

Communication using speech inherently involves recognition errors. Phones, particularly cellular phones, often pick up background noise, so speech communication must be robust in such environments. Therefore, various studies such as [Hirsch and Pearce, 2000] have focused on improving speech recognition accuracy in noisy situations.

In addition to speech recognition errors, which correspond to the system mishearing the user, the possibility of the user mishearing speech should also be considered. Although the quality of text-to-speech (TTS) systems has improved, the

control of intonation remains a problem. Automatically generating appropriate intonations and accents for individual proper nouns is nearly impossible. Therefore, the expressions a system uses should be intelligible so that the user easily understands them, especially when TTS systems are used.

We developed a method of automatically adding appropriate expressions to system responses that makes hearing content words including proper nouns easier. We aimed to reduce users mishearing and to improve the certainty of speech communications, even in noisy situations.

To clarify a confusing spelling, English speakers often use the phonetic code, e.g., “A for alpha”, “B for boy”, and “C for Charlie”. Because we think that the phonetic code is not intuitive enough, we also used words or expressions that explain larger units than phones to indicate differences between confusing words. We used multiple knowledge sources (dictionaries) to automatically obtain candidate words to be added to confirmations as clarifying expressions.

Criteria are needed to select the best clarifying expression among candidates. We define the ease in hearing the clarifying expression using two criteria: the uniqueness of the clarifying expression and its difference from the original expression. By comparing weighted sums of the two criteria for each candidate, the best clarifying expression is selected and spoken to users. The candidates are automatically extracted from knowledge sources that are generally used, and the best candidate is automatically selected. This means that our method does not depend on a specific domain, so it can be applied to other domains without new generation rules.

## 2 Design of System Responses to Avoid User Misunderstanding

### 2.1 Misunderstanding Caused by Phonologically Similar Words

In communications using speech media, speech recognition errors cannot be avoided. To successfully communicate, user mishearing of the system's responses must be avoided, as must system recognition errors. In particular, errors in communicating content words are fatal in spoken dialogue systems because such words are used as keywords for queries. To avoid such errors, many systems require confirmation of content words [Hazen *et al.*, 2000]. The confirmation, however, may not be understood when phonologically confusing

---

S1: Please tell me your current bus stop, destination, or specific bus route.  
U1: From *Kinkaku-ji* (金閣寺) temple.  
S2: Did you say “from *Kinkaku-ji* (金閣寺) temple”?  
U2: ??? (The user cannot discern whether the system said *Kinkaku-ji* or *Ginkaku-ji*.)

---

Figure 1: Example dialogue from conventional system.

---

S1: Please tell me your current bus stop, destination, or specific bus route.  
U1: From *Kinkaku-ji* (金閣寺) temple.  
S2: I am not sure whether you said *Kinkaku-ji* (金閣寺) temple or *Ginkaku-ji* (銀閣寺) temple. Does what you said begin with the character “*kin* (金)”, which means gold in English?  
U2: Yes.

---

Figure 2: Example dialogue from our system.

words are used. In such cases, humans can distinguish such words by focusing on the confusing part and adjusting the intonation, accent, speed, etc. For the systems, automatically giving proper nouns the proper intonation and accent is nearly impossible, just as non-native speakers have difficulty pronouncing unknown proper nouns correctly. Therefore, intelligible language expressions of the confirmations are needed as well as precise TTS engines to correctly transmit the system responses.

Our goal was to prevent the mishearing of content words in spoken dialogue systems. When the content words in confirmation responses are phonologically confusing, our system automatically adds expressions to clarify the words. The system automatically extracts phonologically similar word pairs from the vocabulary and generates clarifying expressions by using multiple knowledge sources. Clarifying expressions make confirmations intelligible, but may also make dialogues lengthy. What words will be added to the clarifying expressions can be controlled by setting a threshold to the phonetic distance between confusing words.

An example dialogue from a conventional system is shown in Figure 1, and one from our system is shown in Figure 2. Because phonologically confusing words are in the systems’ vocabularies such as *Kinkaku-ji* and *Ginkaku-ji*, which are names of temples, the confirmation “Did you say from *Kinkaku-ji* temple?” in Figure 1 may sound similar to “from *Ginkaku-ji* temple”, especially when the quality of the TTS is low or is listened to in a noisy environment. Therefore, our system generates a confirmation that contains an expression that clarifies the difference between phonologically confusing words (e.g., “Does what you said begin with the character *kin* (金), which means gold in English?” in Figure 2).

## 2.2 Generating Confirmation without Dependence on Domains

Our goal is to generate the kind of confirmation described in the previous subsection without dependence on any specific domain. That is, we do not manually describe rules for individual words in a system’s vocabulary, but exploit existing knowledge sources such as dictionaries. This enables us to automatically apply our method to various systems.

Specifically, our system extracts word pairs that are phonologically similar from a system’s vocabulary. It also automatically obtains candidates for clarifying expressions from dictionaries to distinguish the pairs. As a unit to explain the difference between confusing pairs, we adopted Chinese characters<sup>1</sup> because we think a larger unit than a phone is more intuitive. We exploit existing dictionaries that explain the unit, such as a set of words consisting of several Chinese characters, a Japanese-English dictionary, a dictionary describing readings of Chinese characters, and the Japanese phonetic code<sup>2</sup>. None of these dictionaries were constructed by hand for our system but generally used. The system automatically selects the best candidate expression based on its intelligibility. Thus, the clarifying expressions are generated automatically.

## 2.3 Intelligibility of System Responses

We assume that the intelligibility of system responses consists of two factors: difficulty and ease in hearing.

The difficulty indicates whether users understand the meaning instantly upon hearing an expression. Therefore, the intelligibility of a system response is improved when an expression with low difficulty is used. We define the difficulty of an expression by the difficulty of its words. We incorporated the difficulty by using dictionaries in which difficulty rankings are given.

The ease in hearing indicates whether the expression is confusing. Therefore, it is defined by considering both whether the expression is generally similar to other expressions and how the response changed phonologically from the original confusing response by adding the clarifying expression. We define the former as the uniqueness and the latter as the difference of a clarifying expression. The ease in hearing is defined by the weighted sum of the two criteria.

## 3 Generating Clarifying Expressions for Phonologically Similar Words

We describe our method for generating clarifying expressions by obtaining distinctive words from dictionaries. Confusing word pairs are automatically extracted from the system vocabulary and are distinguished using the distinctive words. The procedure is listed below and depicted in Figure 3.

1. We define the phonetic distance between two words.  
By calculating the distance for all the combinations in

<sup>1</sup>Because Chinese characters are ideograms, Japanese people often tell the difference between two words using the meanings of the characters.

<sup>2</sup>The Japanese phonetic code describes each Japanese syllable.

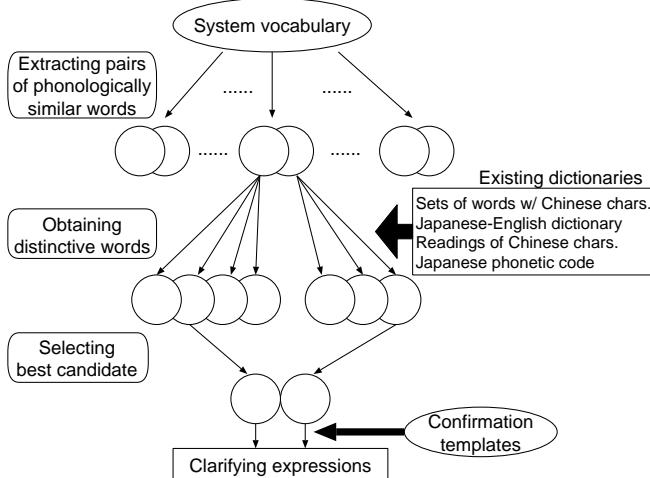


Figure 3: Flow to generate clarifying expressions.

the system’s vocabulary<sup>3</sup>, phonologically similar word pairs are extracted.

2. Distinctive words that include different spelling or phones between the phonologically similar pair are obtained for each pair from dictionaries. The words are candidates for use in clarifying expressions.
3. The best candidate among the distinctive words is selected. An expression of confirmation is generated by applying the candidate to templates for response sentences.

We describe the details of each step in the following sections.

### 3.1 Extracting Pairs of Phonologically Similar Words from System Vocabulary

We describe our method of extracting phonologically similar word pairs from a target system’s vocabulary. First, we define the phonetic distance between words, which represents how phonologically similar two words are. Two words are phonologically similar, that is, confusing, if they have more common phonemes and less different phonemes. Furthermore, in Japanese, because words are pronounced by following a definite rhythm called mora<sup>4</sup>, two words having the same number of morae tend to be confusing.

Considering these characteristics, we define a phonetic distance,  $p.d.(w, w')$ , between  $w$  and  $w'$  as follows:

$$p.d.(w, w') = \frac{e.d.(w, w')}{\log(\min(|w|, |w'|) + 1)}, \quad (1)$$

where  $|w|$  denotes the number of phonemes in word  $w$ . The

<sup>3</sup>Although the cost to compute the distance between all word pairs is high, it can be done beforehand.

<sup>4</sup>A mora usually consist of a consonant and a vowel. Since Japanese is a mora-timed language, the rhythm of Japanese places a beat on each mora in words, which is called mora isochronism.

Table 1: Examples of phonologically similar word pairs.

<i>kyotofuritsu mogakko</i> (京都府立盲学校) (Kyoto prefectural school for the blind)	0.34
<i>kyotofuritsu rogakko</i> (京都府立聾学校) (Kyoto prefectural school for the deaf)	
<i>rakusai koko mae</i> (洛西高校前) (In front of Rakusai high school)	0.36
<i>rakusui koko mae</i> (洛水高校前) (In front of Rakusui high school)	
<i>kinkaku-ji michi</i> (金閣寺道) (Kinkaku-ji temple road)	0.37
<i>ginkaku-ji michi</i> (銀閣寺道) (Ginkaku-ji temple road)	

numerator,  $e.d.(w, w')$ , denotes the edit distance<sup>5</sup> between the phoneme sequences  $w$  and  $w'$ . The edit distance is calculated using DP matching [Navarro, 2001]. In calculating an edit distance, a cost is introduced for each insertion, deletion, and substitution to reflect the mora structure. Here,  $C_1$  is a set of vowels, choked sounds, and syllabic nasals, and  $C_2$  is a set of consonants and semi-vowels. If edited phonemes belong to  $C_1$ , the cost is doubled because the number of morae is changed.

The system extracts word pairs whose phonetic distances are low as phonologically similar words. Examples of such word pairs in the Kyoto City Bus Information System are shown in Table 1.

### 3.2 Obtaining Candidates for Clarifying Expressions

We describe how to obtain distinctive words as candidates for clarifying expressions to distinguish phonologically similar word pairs, which was described in the previous section. We focus on differences of phones or spelling differences between the pairs. The system obtains distinctive words that include the different spelling or phones from multiple knowledge sources. We used four the knowledge sources described later.

First, the system extracts characters, which are Chinese characters here, or phones that are not common between confusing word pairs. Hereafter, the first character or phone extracted is denoted as  $\langle \text{char} \rangle$  or  $\langle \text{phone} \rangle$ <sup>6</sup>. Their positions in confusing words are denoted as  $\langle \text{charLoc} \rangle$  and  $\langle \text{phoneLoc} \rangle$  represented using common characters (or phones) before the difference. An example is shown below for *rakusai koko mae* (洛西高校前) and *rakusui koko mae* (洛水高校前).

$$\begin{cases} \langle \text{char} \rangle = “\text{西}, “\text{水}” \\ \langle \text{phone} \rangle = “\text{sa}\text{'}, “\text{su}\text{'”} \\ \langle \text{charLoc} \rangle = “\text{after the character raku (洛)}” \\ \langle \text{phoneLoc} \rangle = “\text{after the phones raku}” \end{cases} \quad (2)$$

When the difference is at the beginning (or end) of a word, we use  $\langle \text{charLoc} \rangle = \langle \text{phoneLoc} \rangle = “\text{at the beginning}”$

<sup>5</sup>The edit distance of two strings,  $s_1$  and  $s_2$ , is defined as the minimum number of mutations to change  $s_1$  into  $s_2$ , where a mutation is a substitution, insertion, or deletion.

<sup>6</sup>Phones in Japanese correspond to the Japanese syllabary.

Table 2: Specifications of JC1.

difficulty	# words	example
A1 (low)	2,337	反对 (objection)
A2	2,618	主要 (main)
B	4,892	警告 (admonition)
C	8,989	歛待 (hospitality)
F (high)	16,239	違憲 (unconstitutionality)

合図 (sign), 金曜 (Friday), 空気 (air),  
水曜日 (Wednesday), 反対 (objection), ...

Figure 4: Examples of pairs of Chinese characters.

Chinese character	native Japanese readings
外 (out)	soto, hazusu, hoka
音 (sound)	oto, ne
声 (voice)	koe, kowa
読 (read)	yomu

Figure 5: Examples of Japanese readings of Chinese characters.

(or “at the end”).

The system obtains distinctive words from knowledge sources as candidates that contain the different spelling or phones. We will describe four methods corresponding to the knowledge sources used.

### Method Using Words Consisting of Several Chinese Characters

The system uses words consisting of several Chinese characters including differences in spelling. We used a dictionary, JC1 [Sato, 2004], in which the difficulty rankings are given. The specifications of JC1 are listed in Table 2. We used words having the lowest difficulty to improve the intelligibility of expressions. This dictionary includes 2,337 words with the lowest difficulty, part of which is shown in Figure 4. The template to generate expressions of confirmation is as follows:

“Does the character <char> in <ccpair> follow <charLoc>?”

Here, <ccpair> denotes a word in the dictionary. Inappropriate words are removed from the candidates; for example, when a word extracted from the dictionary is a part of its original confusing word pairs, etc.

### Method Using Readings of Chinese Characters

We used Japanese readings of Chinese characters to explain confusing words. We adopted Japanese readings if the target Chinese character had other Japanese readings. Examples of the set of readings are shown in Figure 5. The template to generate clarifying expressions of confirmation is as follows:

“Does the character read as <reading> follow <charLoc>?”

Here, <reading> denotes another reading of the target Chinese character.

Chinese character	English translation
雨	rain
黄	yellow
石	stone
橋	bridge

Figure 6: Examples of translations of Chinese characters.

phone	description word
sa	sakura (cherry blossom)
shi	shimbun (newspaper)
fu	fujisan (Mt. Fuji)

Figure 7: Part of Japanese phonetic code.

### Method Using English Translation of Chinese Characters

We used English translation of a Chinese character when a single Chinese character had a meaning by itself. If a Japanese-English dictionary has an index entry that consists of a single Chinese character, we assume the entry has a meaning itself, and it is used with its English translation. We obtained 148 single Chinese characters and their translations, after limiting them to those with the lowest difficulty. Examples of the set of single Chinese characters and their translations are shown in Figure 6. The template to generate expressions of confirmation is as follows:

“Does the character <char>, which means <trans> in English, follow <charLoc>?”

Here, <trans> denotes a translation of a target character.

### Method Based on Phonetic Code

We also used the Japanese phonetic code to explain phones at the beginnings of differences. A part of the table is shown in Figure 7. We used a word in the table as a distinctive one to explain a phone. The template to generate expressions of confirmation is as follows:

“Does the phone <phone> of <descword> follow <phoneLoc>?”

Here, <descword> denotes a word describing a phone in the Japanese phonetic code.

### 3.3 Selecting Best Candidates for Clarifying Expressions

We describe how to select the most intelligible expressions from candidates. As described in Section 2.3, intelligibility depends on both the difficulty of words and the ease in hearing. To reduce the difficulty, we only used words having the lowest difficulty rankings when obtaining candidates from dictionaries, as described in the previous section.

The ease in hearing is defined by the uniqueness of the distinctive word in the clarifying expression and the difference between the distinctive and original words. Here, we define the uniqueness,  $u_i$ , and the difference,  $d_i$ , of distinctive word  $w_i$ .

The uniqueness,  $u_i$ , represents how words are included that are phonologically similar to word  $w_i$  in the system’s

Table 3: Average number of candidates and percentage of times that at least one candidate was obtained from sources (bus system).

average # candidates	percentage of times candidate was obtained			
	I	II	III	IV
3.9	44%	46%	13%	100%

- I using words with several Chinese characters
- II using readings of Chinese characters
- III using translation of Chinese characters
- IV using phonetic code

Table 4: Distribution of subjective labels of clarifying expressions (bus system).

	A	B	C
labels for all obtained candidates	113	102	13
labels for selected candidates	49	9	0

knowledge sources, such as sets of words consisting of several Chinese characters, a Japanese-English dictionary, and the Japanese phonetic code. The user will not confuse the word with others when few similar words are in the knowledge sources.

The uniqueness,  $u_i$ , of word  $w_i$  is defined as follows:

$$u_i = \frac{1}{N} \sum_{k=1}^N p.d.(w_i, \tilde{w}_k),$$

where  $p.d.(w_i, \tilde{w}_k)$  denotes the phonetic distance defined in Section 3.1. Words having the  $k$  smallest  $p.d.(w_i, \tilde{w}_k)$  in all knowledge sources are denoted as  $\tilde{w}_k$ . That is,  $u_i$  is defined as the average of the  $N$  smallest phonetic distances of words. Here, we set  $N = 10$ .

The difference represents how a distinctive word to be used in a clarifying expression differs from the original one. The clarifying expression is needed when the original expression is phonologically confusing. Therefore, it may still be confusing if the distinctive word is similar to the original one. The difference represents the intelligibility when original words are presented.

The definition of the difference,  $d_i$ , of distinctive word  $w_i$  is as follows. Here,  $w_{long}$  and  $w_{short}$  denote the words with more and less phonemes of the original word,  $w$ , and distinctive word  $w_i$ . The number of phonemes of  $w_{long}$  and  $w_{short}$  are  $N_l$  and  $N_s$ . The difference,  $d_i$ , is defined as the minimum of the phonetic distance between  $w_{short}$  and all partial phoneme sequences in  $w_{long}$  whose lengths are  $N_s$ :

$$d_i = \min_{k=0, \dots, N_l - N_s} \{p.d.(w_{long}[k+1 \dots k+N_s], w_{short})\},$$

where  $w[k \dots k']$  denotes a partial sequence in  $w_{long}$  from the  $k$ -th phoneme to the  $k'$ -th phoneme.

Using the above two criteria, we assume that the more intelligible candidates have larger values of uniqueness and difference. We define a score representing the intelligibility of the clarifying expression including distinctive word  $w_i$  as the weighted sum of the uniqueness and the difference. The system compares the scores and selects the candidate that has the largest one.

$$score(w_i) = W_1 \cdot u_i + W_2 \cdot d_i$$

Table 5: Average number of candidates and percentage of times that at least one candidate was obtained from sources (hotel system).

average # candidates	percentage of times candidate was obtained			
	I	II	III	IV
3.3	40%	48%	15%	96%

Table 6: Distribution of subjective labels of clarifying expressions (hotel system).

	A	B	C
labels for all obtained candidates	102	60	9
labels for selected candidates	47	5	0

We set the weights as  $W_1 = 4$  and  $W_2 = 1$  after examining various values in a preliminary experiment<sup>7</sup>.

## 4 Experimental Evaluation

### 4.1 Implementation into Existing Systems

We applied our method to the vocabulary of the Kyoto City Bus Information System [Komatani *et al.*, 2003]. The vocabulary includes the names of bus stops, bus route numbers, and names of famous places or public facilities near bus stops. The vocabulary size is 1,574.

Table 3 shows the average number of candidates per confusing pair and the percentage of times that at least one candidate was obtained from knowledge sources, when our method was applied to the vocabulary of the bus system. Candidates for clarifying expressions were not always obtained because each knowledge source may not a word including different characters or phones between confusing word pairs. The method using the phonetic code can generate candidates for all words because it focuses on phones and distinctive words are prepared for all phones in the Japanese phonetic code. As a result, on average, 3.9 candidates were obtained per confusing pair from the knowledge sources.

We evaluated whether the generated clarifying expressions were intelligible or not. We gave labels subjectively for each expression: from A (intelligible) to C (unintelligible). The distributions of the subjective labels for candidate expressions and those of the selected expressions with maximum scores are shown in Table 4. This table indicates that expressions with better subjective labels were mostly chosen.

We also applied our method to another system: a hotel search system [Komatani *et al.*, 2001]. The vocabulary includes 865 names: those of wider areas than the bus system uses, facilities of hotels, etc. In this domain, candidates were generated from each knowledge source, and 3.3 were generated on average as shown in Table 5. The selection module also functions well without modifications, as shown in Table 6.

### 4.2 Listening Experiment for Generated Confirmations

We also evaluated whether generated confirmations actually improved the ease in hearing the difference between phono-

<sup>7</sup>Although we set the values based on the bus system's vocabulary, it worked well in the hotel system described later.

Table 7: Results of conventional confirmation.

Subject	Correct	Incorrect	Cannot discern
A	21	0	34
B	38	8	9
C	23	1	31
D	25	0	30
E	32	0	23
Average	27.8 (51%)	1.8 (3%)	25.4 (46%)

Table 8: Results of confirmation using clarifying expressions.

Subject	Total	Correct	Incorrect	Cannot discern
A	34	26	0	8
B	9	5	2	2
C	31	14	3	14
D	30	21	0	9
E	23	22	1	0
Average	25.4	17.6 (69%)	1.2 (5%)	6.6 (26%)

logically confusing word pairs. The procedure of our experiment is as follows. Here, we denote each phonologically similar word pair as  $word_1$  and  $word_2$ . First, a user was displayed  $word_1$  as text. Second, he/she listened to the following confirmation generated by a TTS system, and answered “yes”, “no”, or “I cannot tell”.

I am not sure whether you said  $word_1$  or  $word_2$ .  
Did you say  $X$ ?

where  $X$  was either  $word_1$  or  $word_2$ , determined at random. Then, if his/her answer was “I cannot tell”, he/she also listened to the following confirmation generated by a TTS system, and answered similarly:

I am not sure whether you said  $word_1$  or  $word_2$ .  
(confirmation with clarifying expressions for  $X$ )

He/She could listen to each confirmation up to three times.

We added background noise to the TTS sound, because we assume that the system will be used in noisy environments. The volume of noise was set to equal that of the TTS sound, and we used background noises from shopping malls. Five subjects listened to 55 confusing word pairs that had been automatically extracted from the bus and hotel system vocabularies.

The results of the first confirmation for confusing word pairs “Did you say  $X$ ?” are shown in Table 7. Almost half the word pairs could not be discerned although users were allowed to listen up to three times. Word pairs that could not be discerned in the first confirmation were confirmed again using the clarifying expressions. The results are in Table 8 show that 69% of the word pairs that could not be discerned in the first confirmation became discernible using the confirmations with clarifying expressions. That is, our method improved the intelligibility of confirmations for phonologically similar word pairs that could not be discerned using conventional confirmations. However, 26% of the word pairs still remained unintelligible after the confirmations using clarifying expressions. One of the reasons for this is that the distinctive words used in the clarifying expressions semantically had nothing to do with the context and could not be expected by users.

## 5 Conclusion

We described a method of generating clarifying expressions to prevent confusion between phonologically confusing word pairs in system vocabularies. A framework of our method can be applied to various domains because confusing pairs are extracted automatically and candidates for clarifying expressions are automatically generated and selected. Therefore, we can generate clarifying expressions without preparing them for each domain. We experimentally verified that clarifying expressions could improve listener abilities to discriminate between phonologically similar words.

Because our target language was Japanese, we adopted Chinese characters as units and generated clarifying expressions to explain them. If appropriate morphemes can be defined for other languages, phonologically confusing word pairs can be distinguished by preparing knowledge sources including the morphemes. This would be helpful even when users cannot accurately spell words.

The intelligibility of clarifying expressions also depends on user characteristics and the context. The distinctive words may be unrelated to the contexts. The intelligibility will improve further if such words were more familiar, which depends on individual users. Confirmations for the remaining 31% of words that could not be discerned in our experiment would be more intelligible if clarifying expressions were generated adaptively for individual users.

## References

- [Hazen *et al.*, 2000] T. J. Hazen, T. Burianek, J. Polifroni, and S. Seneff. Integrating recognition confidence scoring with language understanding and dialogue modeling. In *Proc. ICSLP*, 2000.
- [Hirsch and Pearce, 2000] H.-G. Hirsch and D. Pearce. The AURORA experimental framework for the performance evaluation of speech recognition under noisy conditions. In *Proceedings of the ISCA ITRW ASR*, 2000.
- [Komatani *et al.*, 2001] K. Komatani, K. Tanaka, H. Kashima, and T. Kawahara. Domain-independent spoken dialogue platform using key-phrase spotting based on combined language model. In *Proc. EUROSPEECH*, pages 1319–1322, 2001.
- [Komatani *et al.*, 2003] K. Komatani, S. Ueno, T. Kawahara, and H. G. Okuno. Flexible guidance generation using user model in spoken dialogue systems. In *Proc. ACL*, pages 256–263, 2003.
- [Navarro, 2001] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.
- [Raux and Eskenazi, 2004] A. Raux and M. Eskenazi. Non-native users in the let’s go!! spoken dialogue system: Dealing with linguistic mismatch. In *Proc. of HLT/NAACL*, 2004.
- [Sato, 2004] S. Sato. Identifying spelling variations of Japanese words (in Japanese). In *IPSJ SIG Technical Report*, 2004-NL-161, pages 97–104, 2004.

# Towards a Decent Recognition Rate for the Automatic Classification of a Multidimensional Dialogue Act Tagset

Stephan Lesch and Thomas Kleinbauer and Jan Alexandersson\*

DFKI GmbH

Stuhlsatzenhausweg 3, D-66123 Saarbrücken

{lesch, kleiba, janal}@dfki.de

## Abstract

In this paper, we present some thoughts and examinations on statistical dialogue act annotation using multidimensional dialogue act labels, based on the ICSI meeting corpus and the associated MRDA tag set. We show some statistics of this corpus, and preliminary results of a statistical tagger for the dialogue act labels, together with a proposal for a more realistic interpretation of these results.

## 1 Introduction

A crucial capability of automatic speech processing systems is to determine the type of an utterance – question or statement or backchannel, etc. A common way to formalise this kind of information is to compile a categorisation of *dialogue acts* [Austin, 1962; Searle, 1969] into a set of tags that meets best the requirements of the underlying task. With such a tagset it is possible to annotate a corpus of sample dialogues which can then be used as training material for a statistical classifier.

The ICSI<sup>1</sup> meeting recorder project [Dhillon *et al.*, 2004], has developed a corpus containing roughly 72 hours of recordings of actual meetings. The corpus is fully annotated with a multidimensional tagset, which we will refer to as the MRDA tagset in this paper. A dialogue act in the MRDA set consists of a general tag, e.g. *statement* (s) and up to seven special tags that provide additional facets. For example, the label qy^rt stands for *yes-no question with rising tone*.

A straight-forward way to use the MRDA tagset for automatic recognition would be to treat each possible label as a monolithic unit, i.e. ignore the underlying multidimensional structure and instead understand a label merely as a string of characters. Then, after choosing a set of features and training the classifier, one can evaluate the quality of the classifier using traditional metrics like e.g. recall and precision.

Such a view, however, implies discarding useful structural information for both the classification process as well as for the evaluation. It is clear for instance that the dialogue acts qy and qy^rt are related. Therefore, if a qy^rt-utterance

is misclassified, it makes a difference if it was classified as qy or as s - the latter did not even get the general tag correct. This effect is not reflected by traditional recall and precision measures where a classification is either correct or incorrect. Conversely, one expects an informed classifier which utilises the multidimensional properties of the MRDA tagset to yield better recognition rates than one that does not.

To verify this hypothesis, we take a closer look at the ICSI corpus. An initial investigation shows that only 82 labels occur more than 100 times and that the vast majority of the total 2050 labels occur just a few times (see figure 1). Consequently, it is very hard to use these rare acts for classification.

rank	dialogue act	count	percent
1	s	25684	23.03
2	b	14467	12.97
3	fh	6160	5.52
4	s^bk	5674	5.08
5	s^aa	4626	4.15
:	:	:	
29	b.%	511	0.46
30	%	460	0.41
:	:	:	
42	h	263	0.24
:	:	:	
50	h s	193	0.17
:	:	:	
83	s^m	100	0.09
:	:	:	
1057	qy^bu^cs^d^rt	2	0.000018
1058	s^ar^bd %	1	0.000009
:	:	:	
2049	qy^q^cs^d^rt	1	0.000009
2050	s:s^bk s^rt	1	0.000009

Table 1: An excerpt from the dialogue act frequencies for the ICSI meeting corpus (Version 040317).

\*The research presented here is funded by the EU under the grant FP6-506811 (AMI).

<sup>1</sup>International Computer Science Institute at Berkeley, CA

We have made some preliminary classification experiments and trained a maximum entropy classifier using 20000 utter-

ances from the corpus and different variations of the tagset. This classifier was tested on a set of 14512 different utterances. We achieved 51.3% correct classifications. However, a more detailed analysis of the classification results reveals that there are another 20.2% of classifications which are assigned a less specific label, i.e., the correct general tag, but some special tags are missing. Additionally, 3.6% of the classifications are too specific, i.e., some special tags were assigned which are not present in the human annotation. Another 5.8% were “neighbours”, which means they share a common supertype (for instance, the general tag) with the correct label.

We conclude that there is on the one hand room for improvements of the classification and the metric for evaluation could be developed to account for the “almost-hits”.

The paper is organised as follows: the next section describes the MRDA tagset and a simplification thereof—the MALTUS tagset. In section 3, we discuss some of the characteristics of the ICSI meeting corpus and show how a classifier improves as the amount of training data increases. Section 4 details the measures used for the evaluation of classifiers and proposes a new measure. The next section describes the classification experiments. Finally, in section 6 we conclude the paper and provide some future directions.

## 2 Multidimensional Tagsets

The labels of a dialog act tagset are not necessarily multidimensional. The Verbmobil System, for example [Alexandersson *et al.*, 1998], used a small set of roughly 30 tags tailored to its particular application, the automatic translation of telephone negotiations. Examples of the Verbmobil tags are greet, bye, introduce, request, suggest.

Multidimensional tagsets, on the other hand, allow to annotate several aspects of an utterance. The DAMSL<sup>2</sup> tagset, for instance, defines four aspects: the communicative status, the information level and the forward and backward looking function of the utterance. A variant of the DAMSL tagset, the SWBD tagset [Daniel Jurafsky, 1997], was used for annotation in the Switchboard project; the SWBD tagset, in turn, served as the basis for the MRDA tagset [Popescu-Belis, 2003].

### 2.1 The MRDA Tagset

The “Meeting Recorder Dialogue Act” tagset was used to annotate the ICSI meeting corpus.<sup>3</sup> Labels consist of a general tag, which may be followed by one or several special tags and a disruption mark, or of a disruption mark only. The general form is

(<general tag>(<special tag>)?) (.<disruption mark>)?

with the following tags:

- General tags are statement (s), questions (qy/qw/qr/qrr/qo/qh), backchannel (b) and floor management (fg/fh/h).
- There are 40 special tags describing backchannels, positive, negative or uncertain responses, restatements (repe-

<sup>2</sup>Dialogue Act Markup on Several Layers, [Allen and Core, 1997]

<sup>3</sup>See <http://www.icsi.berkeley.edu/Speech/mr/>

titions or corrections), politeness mechanisms and other functions.

- Disruption forms are “interrupted by other speaker” (%–) and “abandoned by speaker” (%––). Two other tags, “indecipherable” (%) and “non-speech” (x), are included in this group.

Furthermore, there are two kinds of *compound labels*. Some utterances consist of two closely adjoining parts which constitute two DAs: e.g., a floor grabber followed by a statement can be annotated by a compound label fg|s. The other case is quoted speech, where labels are combined using a colon (e.g. s:s).

### 2.2 The MALTUS Tagset

MALTUS, introduced in [Popescu-Belis, 2003], is an attempt to abstract from the MRDA tagset in order to reduce the huge number of possible labels. Several groups of MRDA tags were grouped into one MALTUS tag, and some MRDA tags were dropped altogether. An utterance is marked either as uninterpretable (U), or with one general tag (tier 1 tag, T1) and zero to five special tags (tier 2 tags, T2). Also, a disruption mark (D) may be appended. The general form of a MALTUS label is

(U | T1 ^ T2)? (.D)?

with the following tags:

- tier 1 tags are statement (S), questions (Q), backchannel (B) and floor holder (H).
- tier 2 tags are response types (RP/RN/RU) attention (AT), actions (DO), restated information in corrections or repetitions (RIC/RIR) and politeness (PO).

## 3 Some Corpus Characteristics

The experiments presented are based on the the ICSI meeting corpus [Janin *et al.*, 2003], a collection of 75 meetings of roughly one hour each.

The corpus is available as text files. Each line describes one utterance: the transcribed text, the start and end times of the utterance, the time alignments of each word in the transcription, the DA label, the channel name and (optionally) adjacency pair annotation. However, the files do not contain syntactical or semantic information, POS tags or any phonological features.

The MRDA tagset theoretically allows up to several million different labels, but only some thousand of them actually occur in the corpus: the 04/03/17 version of the corpus contains 112027 utterances with 2050 different DA labels. Some of these labels are compound labels of the form a|b; we split these utterances and obtain 118694 utterances with 1256 different labels. Some utterances are explicitly marked as non-labelled (z), and some are not labelled at all; these utterances and their successors are ignored, leaving 116097 utterances from which we take the training and testing material.

### 3.1 Distribution of general categories over the ICSI corpus

When we map the MRDA labels to the five basic categories (statements, questions, backchannels, floor management and

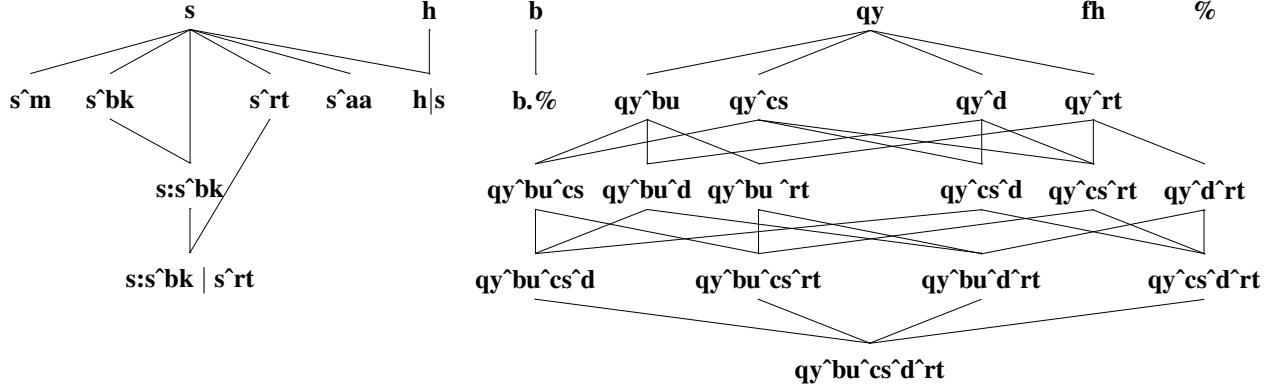


Figure 1: The lattice formed by the MRDA labels shown in table 1. Labels are ordered by the subset relation. *Compound labels*, i. e., , two labels combined with “|” or “:”, are daughters of the two separate labels. Note that only the parts of the compound labels | were used in the classification experiments.

disruptions) in what we call “classmap 1”, we see that the frequencies of these categories are very unevenly distributed - statements make up more than half of the material (See table 2). Note the descending order in the number of training examples for statements, backchannels, floor managements and questions, and how this order is reflected in the recall for these classes in a five-way classification experiment using classmap 1, see figure 4.

Category	gen. tag	%	classm.1	%
Statement	76073	64.09	66640	56.14
Backchannel	15178	12.79	14624	12.32
Floor	12276	10.34	12235	10.31
Question	8522	7.17	7374	6.21
Disruption	4113	3.47	15289	12.88
Z(nonlabeled)	2442	2.06	2442	2.06
X(nonspeech)	90	0.08	90	0.08
$\Sigma$	118694	100%	118694	100%

Table 2: Distribution of the main classes over the corpus.

### 3.2 Words and bigrams

We counted the number of words and bigrams over excerpts from the corpus with different sizes (with 8-fold averaging, using raw words without stemming). The logarithmic plot (see figure 2) shows that the numbers of word and bigram features keep increasing with the number of utterances examined. There is also a constant relation between the number of words and the number of utterance-initial words—there are about five to eight times as many words as initial words. A similar relation holds between bigrams and utterance-initial bigrams.

### 3.3 How much training data do we need for a classifier?

With the specification of a new (MRDA-like) tagset for a corpus of meetings in mind, we were also interested in how much

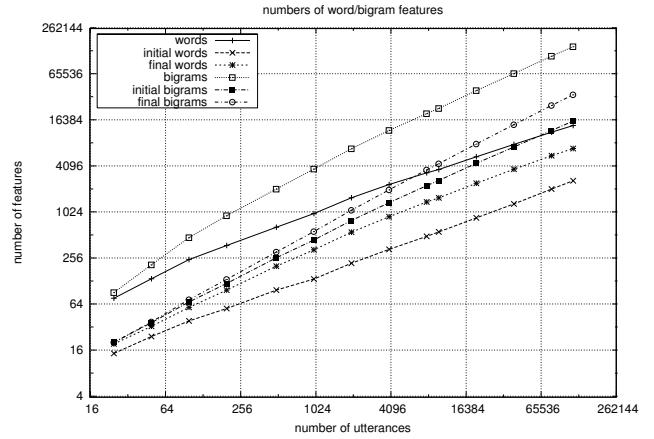


Figure 2: The number of words and bigrams for different numbers of utterances

hand-annotated training material is needed to obtain “decent” classification using a statistical model. We found that the learning curve begins to flatten out at roughly 10000 utterances, but keeps rising with more training data. This observation (see figure 3) holds for the full set of MRDA labels, as well as when we map them to MALTUS labels, or to the five basic classes (using the “classmap 1”).

## 4 A New Metric for the Evaluation of Classification Results

Usually, classification tasks are evaluated using the precision and recall metrics:

$$Precision(l) := \frac{correct(l)}{tagged(l)}$$

$$Recall(l) := \frac{correct(l)}{occurs(l)}$$

where  $occurs(l)$  is the number of times the label  $l$  occurs in the human annotation of the test corpus,  $tagged(l)$  is

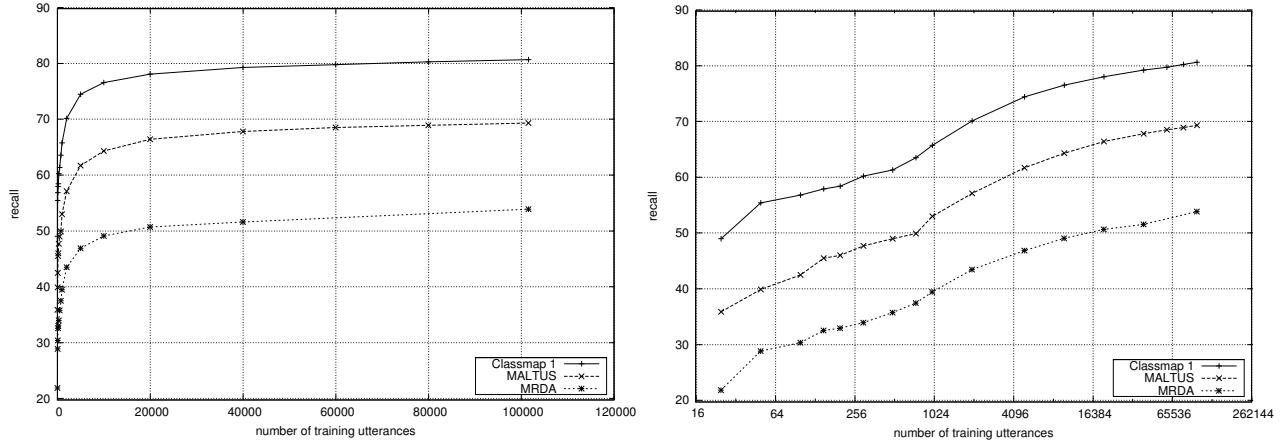


Figure 3: Recall (percent) for MRDA and MALTUS labels, and MRDA mapped with classmap 1, with different sizes of the training set. (linear and log scale, using 4-fold cross-validation, 2-fold for MRDA with 101584 training utterances)

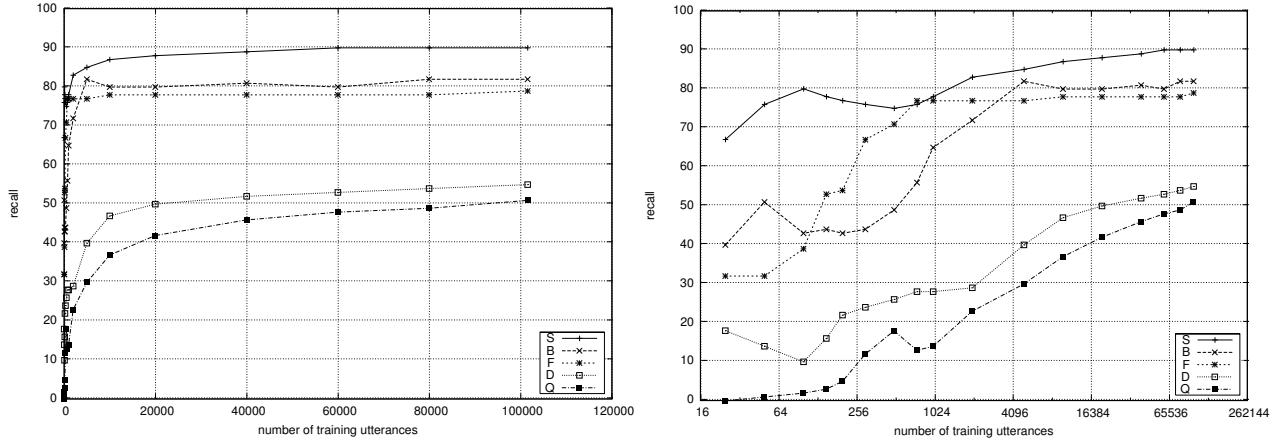


Figure 4: Recall (percent) for statements, questions, floors, backchannels and disruptions (classmap 1, linear and log scale, 4-fold cross-validation)

the number of times it was assigned by the classifier, and  $\text{correct}(l)$  is the number of times it was correctly assigned.

The recall values given in the experiments are the total recall over all labels:

$$\text{Recall} := \frac{\sum_l \text{correct}(l)}{\sum_l \text{occurs}(l)}$$

However, these are binary metrics which do not consider the case that the assigned label is incorrect, but very similar to the correct label. For instance, the label  $s^{\wedge}rt$  marks a statement with rising tone; we can hardly recognise this properly as we do not use phonological features. However, many such utterances will be tagged as  $s$  (statement). By defining a similarity metric between dialogue acts, we can include such cases in the evaluation of the classifier.

One way to define such a similarity metric is to order the labels in a hierarchy according to the sets of tags which make up the labels. For MRDA labels, this means we have several hierarchies with a general tag at the top (see fig. 1). Using

such hierarchies, we can check if the “true” label and the classifier output have a least upper bound (lub). If there is one, there is at least some relationship between the labels. As we found in our experiments, in most cases where the lub exists, the classifier output is underspecific, i.e., some special tags are missing. Using this concept, we define a distance metric between two labels  $DA^T$  (a *true* label) and  $DA^C$  (a *classified* label):

$$\text{SCORRE}(x, y) := \begin{cases} 1 - \frac{\delta^T + \delta^C}{2 \times \text{depth}} & \text{if } DA^{\text{lub}} \text{ exists} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\text{minPath}(x, y) := \text{shortest path between } x \text{ and } y \quad (2)$$

$$\delta^C := |\text{minPath}(DA^C, DA^{\text{lub}})| \quad (3)$$

$$\delta^T := |\text{minPath}(DA^T, DA^{\text{lub}})| \quad (4)$$

For our experiments with MRDA and MALTUS labels, we set *depth* to 5 (with the current ordering of the labels in the ICSI corpus as shown in figure 1, the maximum distance between a *lub* and a label is 5); thus the denominator is 10, and

a SCORRE of 0.9 means that the shortest path between two labels in the hierarchy has length 1.

For a test of a classifier with  $n$  utterances, true labels  $DA_i^T$  and classified labels  $DA_i^C$ , we define

$$\text{SCORRACY} = \frac{\sum_{i=1}^n \text{SCORRE}(DA_i^T, DA_i^C)}{n}$$

We motivate SCORRE by its similarity to *fScore* between two multi-dimensional labels (see also [Lesch *et al.*, 2005]). Considering labels as sets of tags (e.g.  $s \sim rt$  as  $\{s, rt\}$ ) allows us to define precision and recall for a true label  $DA^T$  and a classified label  $DA^C$  by using their intersection. Let

$$DA^I := DA^T \cap DA^C \quad (5)$$

$$\delta^C := |DA^C| - |DA^I| \quad (6)$$

$$\delta^T := |DA^T| - |DA^I| \quad (7)$$

For the normal labels in fig. 1,  $DA^I$  is equivalent to  $DA^{lub}$ , and the set-differences  $\delta^T$  and  $\delta^C$  are equivalent to the distances defined in (3) and (4). Now we can define *precision*, *recall* and *fScore* for a pair of labels  $DA^T$  and  $DA^C$ :

$$\begin{aligned} \text{precision} &:= \frac{|DA^I|}{|DA^C|} = 1 - \frac{\delta^C}{|DA^C|} \\ \text{recall} &:= \frac{|DA^I|}{|DA^T|} = 1 - \frac{\delta^T}{|DA^T|} \\ \text{fScore} &:= \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \\ &= 1 - \frac{\delta^T + \delta^C}{|DA^T| + |DA^C|} \end{aligned}$$

Note the denominators: the distances are normalised to the sizes of the true and the classified labels. Conversely, SCORRE simply normalises to a constant chosen to ensure that it always yields a value between 1 and 0. Consequently, *precision*, *recall* and *fScore* determine which fraction of the output of a classifier is correct, while SCORRE and SCORRACY tell us how much it deviates from the ground truth.

In the following example, testing a classifier on 14512 utterances has resulted in 7823 correct and 4038 approximately correct classifications:

	utterances	$\sum$ Score	avg.
correct	7823	53.9%	7823
approx.correct	4038	27.8%	3542.3
all	14512	100%	11365.3

Since each correct classification contributes 1 to the total SCORRE, and incorrect classifications do not contribute at all, the 4038 approximately correct classifications contribute 3542.3, or 88% on average, i.e., the average distance to the correct label in these cases is 1.

It is clear that this metric is highly dependent on the hierarchy of labels. Measuring the difference between labels by the length of the minimal path between them implies that we consider the edges in the hierarchy as representing equal differences between the content of labels. Without this assumption, one might introduce weights for the edges and define  $\delta^C$  and  $\delta^T$  as the sum of the weights on the cheapest path.

## 5 Classification Experiments

In this section, we report some classification experiments with the complex MRDA/MALTUS labels (that is, without regard to the internal structure of the labels), using an off-the-shelf maximum entropy classifier package for Java.<sup>4</sup>

A maxent model is trained from a set of examples, which consist of the features of an input utterance and its DA label (the class of the input). The resulting model maps (*feature, label*) pairs to weights indicating how strongly the presence of *feature* predicts *label*.

We used the following features:

- word features: the words occurring in the utterance, the initial and final words, and the initial words of the following utterance
- word bigrams: the bigrams occurring in the utterance, and the utterance-initial/final bigrams
- the length of the utterance
- temporal relation features indicating whether there is a pause, no pause or an overlap between the current utterance and the preceding/following one
- features indicating whether the current utterance is the beginning, or ending, or in the middle of a speaker turn
- the DA label of the preceding utterance

Note that some of these features are forward-looking. We would not want to use such features in a dialogue system which is required to react to a user's input; in a meeting-processing application, however, we can expect to be able to use at least the immediate context of an utterance. Note that we did not use any phonological features. Features, like stemming and part-of-speech information would be desirable.

We ran a series of classification experiments using the original MRDA labels, mapping the MRDA labels to MALTUS labels, and finally mapping the MRDA labels to the five categories "statement", "question", "backchannel", "floor management" and "disruptions" (the "classmap 1").

With MRDA and MALTUS labels, we find that only the most frequent labels occur frequently enough to be recognised reliably, or to have a significant influence on testing results.

Out of the 1256 MRDA labels, there are only 80 which occur more than 100 times. However, these 80 labels make up 111496 of all 118694 utterances (94%). There are 265 which occur 10 times or more. This means that about 80% of the labels occur only one to nine times; these labels are almost never correctly recognised. Table 3 shows results of one classification experiment: by simply using the labels as-is, we get approximately 51% correct classifications, and another 29% approximate classifications.

With MALTUS labels, we have significantly less labels (81), and their distribution over the corpus is less uneven: there are 23 labels which occur more than 100 times, and 42 which occur more than 10 times. When we train a classifier for these labels, we see that mostly those which occur more

<sup>4</sup>The Maximum Entropy Classifier by the Stanford NLP Department, available from [nlp.stanford.edu/downloads/classifier.shtml](http://nlp.stanford.edu/downloads/classifier.shtml)

than 100 times are reliably recognised. Table 3 shows the results using the same training/testing set, but with the labels mapped to MALTUS labels. We can see that more utterances are correctly classified (67.1%) than with MRDA labels, and the sum of correct and approximately correct classifications is higher as well. (83.2%).

[Clark and Popescu-Belis, 2004] reports a similar classification experiment without disruption marks and with a slightly different version of the MALTUS tag set and different features, achieving 73.2% accuracy.

event type	MRDA	MALTUS
correct	51.0%	67.1%
overspecific	3.6%	2.7%
underspecific	19.2%	11.2%
neighbour	5.9%	2.1%
approx.correct	28.8%	16.1%
total	79.8%	83.2%

Table 3: Classification results using 20000 utterances as training material and 14512 for testing, 4-fold cross-validation

The maximum generalisation of the tagset which can still be considered useful is to map all labels to one out of five classes: statements, questions, backchannels, floor management and disruptions. (Actually, there is a sixth class, “X” for non-speech noises. However, it is very rare.) We tried two variants of such a mapping:

- One variant (the “classmap 1”) comes with the documentation to the ICSI meeting corpus: this mapping prefers disruptions in some cases - for instance, a disrupted statement is mapped to D, not S. In this case, we only get a recall of 78.7%. A similar result—77.9%—was reported in [Clark and Popescu-Belis, 2004].
- By mapping each label to one of the five classes according to its general tag, we have more instances of statements. The most frequent class which is recognised very well, with a recall of 91%. This leads to an increase of the total recall to 83.8%.
- For a four-way classification experiment—discriminating utterances between statements, questions, backchannels and floor management, and ignoring disruptions—[Clark and Popescu-Belis, 2004] reports 84.9% correct classifications.

### 5.1 An algorithm for the Reduction of the Tagset

The uneven distribution of class frequencies has some disadvantages when we choose to model monolithic labels. The size of the model, and the time required to train it, are rather large, although most of the classes are almost never recognised. Therefore, we used the following approach to reduce the set of classes.

We define the entropy of a set of DA labels and an annotated corpus as

$$H := - \sum_{l \in \text{labels}} p(l) \log_2 p(l)$$

$$p(l) := \frac{\text{number of occurrences of } l}{\text{corpus size}}$$

and for a mother-daughter pair of DAs  $(m, d)$ , the loss in entropy when  $d$  is mapped to  $m$ :

$$\Delta H(m, d) := p(m) \log_2 p(m) + p(d) \log_2 p(d) - (p(m) + p(d)) \log_2 (p(m) + p(d))$$

Then we find the pair  $(m, d)$  in the current set which minimises  $\Delta H$ , and map all occurrences of  $d$  to  $m$ . This step is repeated until the set is reduced to a given size.

This method differs from simply choosing the  $n$  most frequent classes in that it considers collapses the selected pair  $(m, d)$  to  $m$ , no matter which one has the higher frequency (for instance, the label  $qy^r t$  occurs 1022 times,  $qy$  only 368 times). Also, the limitation to mother-daughter pairs means that the labels at the top of a hierarchy (e.g.  $qy$ ) are never removed.

The most frequent classification error is that an instance of a more specific label (e.g.,  $s^blk$ ) is assigned a less specific label ( $s$ ), which is counted as an approximately correct classification. When this pair is collapsed to the less specific one, the same classification would be considered correct. This is what happens when we go from MRDA to MALTUS labels, or even to the 5-way-mapping: we can see a shift from approximately correct to correct classifications, while the sum remains the same or improves slightly (in the range between 80% and 85%).

#das	correct	approx	total	SCORRACY
16	81.5%	0.0%	81.5%	82%
20	73.4%	8.2%	81.4%	81%
25	63.5%	17.7%	81.2%	79%
50	53.4%	27.1%	80.5%	77%
60	52.3%	28.0%	80.3%	77%
70	51.8%	28.4%	80.2%	77%
80	51.6%	28.6%	80.2%	77%
90	51.4%	28.7%	80.1%	76%
100	51.4%	28.8%	80.2%	76%
150	51.3%	28.8%	80.1%	76%
200	51.1%	29.0%	80.1%	76%
300	51.0%	29.1%	80.1%	76%
400	51.0%	28.9%	79.9%	76%
500	51.0%	29.0%	80.0%	76%
750	51.0%	29.0%	80.0%	76%

Table 4: Results (4-fold cross-validation) when the set of MRDA labels is simplified using the entropy-based mapping.

When we use the entropy-based method to define mappings to smaller subsets of the MRDA or MALTUS labels, we observe a similar effect; it only becomes visible when we reduce the set of labels to a very small size (e.g. 25 MRDA or 10 MALTUS labels). We also observe a small improvement in the SCORRE metric. We ascribe this to the uneven distribution of the labels over the corpus. Therefore, this way of shrinking the set of labels does not seem very useful in improving the classification accuracy; however, it significantly reduces the time needed to train a classifier, and the space occupied by the model.

#das	correct	approx	total	SCORRACY
10	71.5%	11.9%	83.4%	82%
20	67.2%	16.1%	83.3%	81%
30	67.1%	16.2%	83.3%	81%
40	67.1%	16.2%	83.3%	81%
50	67.1%	16.1%	83.2%	81%
60	67.1%	16.1%	83.2%	81%
70	67.1%	16.1%	83.2%	81%
81	67.1%	16.1%	83.2%	81%

Table 5: Results (4-fold cross-validation) after mapping MRDA labels to MALTUS labels, and then simplifying using the entropy method. 81 is the full set of labels.

## 6 Discussion and Outlook

We have discussed the task of dialogue act classification for a multidimensional tag set. In particular, we have focussed on the MRDA tag set and the ICSI meeting corpus. We introduced a novel forgiving evaluation metric which utilises a hierarchical view of the tag set. The intuition behind SCORRE is that not hitting the correct tag can be viewed as more or less wrong. We thus depart from the monolithic view of classification results which has been used up until now, e.g., [Reithinger and Klesen, 1997; Stolcke *et al.*, 2000].

We also presented a method to gradually reduce the tag set. We showed that, for our classifier, the overall recognition rate does not change much unless the initial set of labels is reduced drastically, to 50 for the MRDA set, or 10 for MALTUS).

Future work includes the following topics:

### Examining confusion matrices

In our classification experiments based merely on transcriptions of the ICSI meetings, there are some dialogue acts that are often mixed up. In the confusion matrix (table 6), we have highlighted three such dialogue acts:  $s^a a$  (statement and accept),  $s^b b$  (statement and acknowledgement) and  $b$  (backchannel). These acts are among the most frequently confused ones, and have been shown before to be hard to distinguish, e.g., [Reithinger and Klesen, 1997]. This is partly because they share much of their vocabulary (“uhuh”, “yeah”, “right”, “okay”, “absolutely”...). To a degree, they can be distinguished by their acoustic and temporal properties. For instance, accepts and acknowledgements usually occur after another speaker has completed a phrase or utterance, while backchannels can occur in the middle of a phrase of another speaker.

When we find such a pair or group of easily confused labels, we should, on the one hand, try to compare the definitions of these labels, or the tags in them, in order to find new features which we can extract from our training data and which help discriminating between the labels. On the other hand, collapsing these acts would possibly enhance the quality of the classification as well, whereas such a decision has to be taken according to the requirements from the consumers of the classification.

### Classifying aspects separately

In the experiments reported, we train a single classifier for complex labels which are actually combinations of tags representing different aspects of an utterance. This way, most of the rare combinations are nearly impossible to recognise.

A different approach would be to use several separate classifiers, one for each aspect of an utterance. For MRDA labels, we might use one classifier to decide on the general class of an utterance (statement, question, etc.), additional classifiers for groups of tags (e.g., to determine the type of a question), and binary classifiers to check for the presence of independent properties (e.g. rising tone). Using separate classifiers for the different aspects, we might be able to recognise rare combinations of tags more reliably; in particular, it would enable us to recognise combinations which did not occur in the training material.

On the other hand, however, we would lose information about correlations between tags which is included “for free” in a single classifier for the complex labels. In [Clark and Popescu-Belis, 2004], a single classifier for complex MALTUS labels (which reached an accuracy of 73.2%) was compared to a combination of classifiers, which reached only 70.5%.

### Feature analysis

The results in [Clark and Popescu-Belis, 2004] were obtained by using roughly the same kinds of features as in this article—words, bigrams and features indicating the previous dialogue act and temporal overlap between utterances. Especially for words and bigrams, further research is necessary, as their number is almost unlimited. It may prove worthwhile to further investigate to which degree different features add to the overall recognition result. Not only is the memory needed to store these features reduced, the same argument also applies to the time needed to train the classifier. One preliminary result is that ignoring words and bigrams with low frequencies (< 10) has almost no influence on the classification results.

### Adding features

The features we use currently are those which are easy to obtain from the transcriptions available to us; however, they are suboptimal for recognising certain types of utterances. As fig. 4 shows, questions are the type with the worst recall, and we expect an improvement if phonological features were included. Also, we would like to include part-of-speech information.

### Improving the modelling

Although our classifier evaluation takes similarities between labels into account, the maxent classifier package does not. The training procedure classifies the training data according to the current feature weights and adjusts the weights to minimise an error function. This function is based on the number of incorrect classifications and does not recognise partly correct ones. We are going to research whether the quality of the models can be improved by using an error function which is aware of similarities between labels.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Sum
1 qr	1	.	.	.	.	.	.	.	6	.	1	.	.	.	.	1	.	.	.	9	
2 s^aa	.	<b>338</b>	.	.	<b>24</b>	.	.	4	40	62	.	.	.	.	.	12	<b>494</b>	.	.	974	
3 qo	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	1	
4 %	.	2	.	.	2	.	.	11	3	53	.	.	30	1	2	.	3	3	2	.	112
5 s^bk	.	<b>89</b>	.	.	<b>412</b>	.	.	1	36	42	.	.	1	.	.	15	<b>287</b>	.	.	883	
6 qh	1	.	.	.	.	4	.	.	26	.	5	.	.	.	.	5	.	9	.	50	
7 x	.	.	.	.	.	.	.	.	7	.	2	1	.	.	.	.	2	.	.	12	
8 fh	.	7	.	.	3	.	.	659	41	40	.	11	31	3	23	2	1	57	.	.	878
9 fg	.	70	.	.	28	.	.	72	105	16	.	1	14	3	.	.	21	.	.	330	
10 s	1	54	.	.	29	3	.	7	7	6148	.	104	12	.	4	1	37	37	9	57	6510
11 qo^rt	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	1	
12 s.%-	1	.	.	.	.	.	.	1	.	340	.	102	2	.	1	1	3	.	3	454	
13 %-	.	.	.	.	1	.	.	26	6	109	.	12	140	.	8	2	7	2	3	.	316
14 h	.	.	.	.	.	.	.	18	10	1	.	.	19	.	.	.	2	.	.	50	
15 %-	.	1	.	.	.	.	.	26	3	59	.	23	39	.	29	4	.	.	.	184	
16 qrr	.	.	.	.	.	.	.	.	13	.	.	3	.	.	16	1	.	1	.	34	
17 qy	2	5	.	.	4	.	.	1	245	.	10	1	.	.	1	245	47	1	2	564	
18 b	.	<b>78</b>	.	.	<b>89</b>	.	.	2	.	26	.	.	.	.	8	<b>2189</b>	1	.	2393		
19 qw	.	.	.	.	.	.	.	.	47	.	4	1	.	.	2	2	97	.	153		
20 s^df	.	.	.	.	.	.	.	.	447	.	9	.	.	.	3	.	1	144	604		
Sums	6	644	.	.	592	7	.	827	252	7688	.	284	274	27	67	27	344	3143	124	206	
x=y	1	338	.	.	412	4	.	659	105	6148	.	102	140	19	29	16	245	2189	97	144	
x≠y	5	306	.	.	180	3	.	168	147	1540	.	182	134	8	38	11	99	954	27	62	

Table 6: A confusion table for 20 MRDA tags. The labels in the rows are the correct labels, those in the columns are the classifier outputs. E.g., line 2 column 18 (494) means that **s^aa** was misclassified as **b** 494 times—more often than it was correctly recognised.

## References

- [Alexandersson *et al.*, 1998] Jan Alexandersson, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Michael Kipp, Stephan Koch, Elisabeth Maier, Norbert Reithinger, Birte Schmitz, and Melanie Siegel. Dialogue Acts in VERBMOBIL-2 – Second Edition. Verbmobil-Report 226, DFKI Saarbrücken, Universität Stuttgart, Technische Universität Berlin, Universität des Saarlandes, 1998.
- [Allen and Core, 1997] James Allen and Marc Core. Draft of DAMSL: Dialog Act Markup in Several Layers. <http://www.cs.rochester.edu/research/cisd/resources/damsl/> RevisbedManual/ RevisedManual.html, 1997.
- [Austin, 1962] J. L. Austin. *How to do Things with Words*. Oxford University Press, 1962.
- [Clark and Popescu-Belis, 2004] A. Clark and A. Popescu-Belis. Multi-level dialogue act tags. In *Proceedings of SIGDIAL '04 (5th SIGDIAL Workshop on Discourse and Dialog)*, Cambridge, MA., 2004.
- [Daniel Jurafsky, 1997] Debra Biasca Daniel Jurafsky, Elizabeth Shriberg. Switchboard swbd-damsl shallow-discourse-function annotation (coders manual, draft 13). Technical report, University of Colorado, Institute of Cognitive Science, feb 1997. <http://www.colorado.edu/linguistics/faculty/jurafsky/pubs.html#Tech>.
- [Dhillon *et al.*, 2004] Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. Meeting recorder project: Dialog act labeling guide. Technical report, International Computer Science Institute, February 2004. ICSI Technical Report TR-04-002.
- [Janin *et al.*, 2003] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wootters. The ICSI Meeting Corpus. In *Proceedings of ICASSP-2003, Hong Kong*, Hong Kong, 2003 2003. ICASSP.
- [Lesch *et al.*, 2005] Stephan Lesch, Thomas Kleinbauer, and Jan Alexandersson. “a new metric for the evaluation of dialog act classification”. In “*Proceedings of Dialor05, the Ninth Workshop On The Semantics And Pragmatics Of Dialogue (SEMDIAL)*”, 2005.
- [Popescu-Belis, 2003] Andrei Popescu-Belis. Dialogue act tagsets for meeting understanding: an abstraction based on the damsl, switchboard and icsi-mr tagsets. Technical report, ISSCO/TIM/ETI, University of Geneva, September 2003. Version 1.2 (December 2004).
- [Reithinger and Klesen, 1997] Norbert Reithinger and Martin Klesen. Dialogue Act Classification Using Language Models. In *Proceedings of the 5rd European Conference on Speech Communication and Technology (EUROSPEECH-97)*, pages 2235–2238, Rhodes, 1997.
- [Searle, 1969] John R. Searle. *Speech Acts*. University Press, Cambridge, GB, 1969.
- [Stolcke *et al.*, 2000] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech, 2000.

# Efficient Dialogue Using a Probabilistic Nested User Model

Bryan McEleney and Gregory O'Hare

Department of Computer Science  
University College Dublin  
Belfield, Dublin 4  
bryan.mceleney@ucd.ie

## Abstract

We describe a set of dialogue simulation experiments, in which a probabilistic nested user model is employed in deciding between speech acts for a collaborative planning task, finding that a gain in utility can be obtained by using a probabilistic rather than a logical model. Given a set of ordinary dialogue plan rules, our system generates a game-tree representation of the dialogue, using chance nodes to represent uncertain preconditions in the plan. Then, the game-tree is evaluated with respect to a given user model state.

## 1 Introduction

It has long been recognised that generation of cooperative dialogue should involve some model of the hearer. In human dialogue, economy of expression depends on prediction of the interpretation that the hearer will make of a dialogue act, within the context of the dialogue history and the beliefs that support the hearer's inferences. Instead of a reflex response, the hearer produces a deep plan structure that fits the dialogue history [Carberry, 1990], and revises his beliefs based on the preconditions, effects, and their consequences associated with each act in the dialogue history. Then, he adds a consistent act to the inferred plan structure. A number of phenomena arise from this model. Appropriate referring expressions can be selected and interpreted [Heeman and Hirst, 1995]. Indirect speech acts can be planned [Allen and Perrault, 1980]. Extra-cooperative behaviour like correction of misconception [McCoy, 1989] and provision of unasked-for information or acts emerges. With different agents planning each act, it is possible that the contributions that each makes to the overall plan structure are founded on different beliefs about the domain state and the applicable plan rules [Pollack, 1986].

While these phenomena are interesting, there is a problem that has received little attention, and that is to employ plan recognition with nested belief models to generate the shortest possible dialogue. Using each agent's ability to recognise plans, the number and execution time of the dialogue acts can be reduced. This is significant problem in online planning in a spoken language dialogue system, as well as in offline planning of dialogue policies. Such policies can be applied to

human-human interactions, such as deciding whether to ask the customer "Would you like fries with that?", or to graphical or text-based interface design where communicative acts must be chosen. We argue that the way forward in achieving this is to employ a probabilistic nested belief model, rather than a logical one, and in this paper, we show how much more efficient such dialogues can be once a probabilistic model is adopted.

We are currently applying our planner to the problem of collaborative planning, in which agents negotiate over a plan in which both of them will act. This differs from domain-level planning in which the agents act within the object plan. We have also applied our planner in previous work to pure domain-level planning, in which dialogue acts were specified as domain acts. We have used it to decide between using a clarification subdialogue and risking plan failure, which was a problem suggested by [Carletta, 1992], and have shown that the choice of strategy depends on a probabilistic belief variable. A considerable performance gain was obtained by adapting the strategy to this probabilistic variable. We have used it to decide whether to take the initiative in a task [McEleney and O'Hare, 2005], where the first agent must use a model of a second agent to predict whether the second will take the initiative in beginning a collaborative task. Once again, the fact that a probabilistic model was used made a difference to the efficiency of the dialogues. We have also shown that the planner is capable of working with other well known dialogue problems such as correcting misconceptions, and in planning an indirect speech act. Each problem can be easily specified using a nested belief model.

## 2 The planner

Our planner rests on an abstract, game-theoretic model of dialogue, in which agents decide between dialogue strategies in pursuit of rewards, by constructing a game-tree representation of the dialogue. Dialogue is planned to the level of the dialogue act, which is a structure with illocutionary force and a propositional content. There is no translation between the dialogue act and a natural language representation. Since we have no useful data of human dialogues within a collaborative planning domain, and no natural language planner, our system uses a fixed cost estimate for each type of dialogue act, which is adequate to characterise dialogue behaviour, if not to obtain concrete results. The planner ignores the lim-

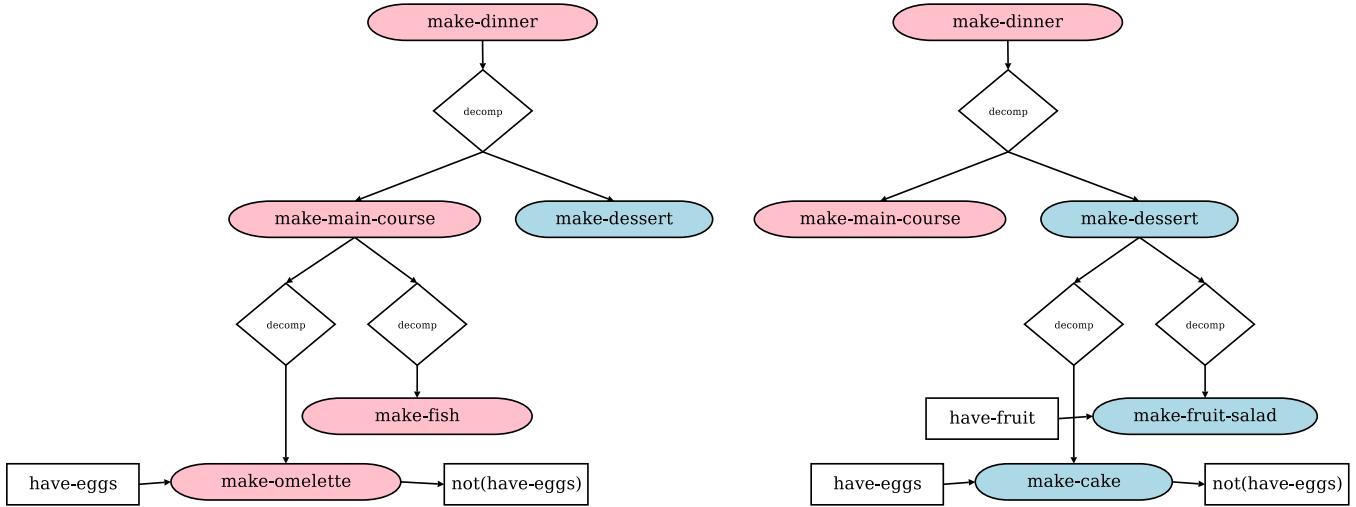


Figure 1: Plan Library

itations of human rationality, and is therefore an unrealistic mechanism for modelling human behaviour, especially as game trees can be large and difficult to construct. Instead we work with perfect utility-maximising players. Nor do we evaluate the planner with system-human trials since a simulation provides much more detailed data about the character of the planning problems that we are investigating. Instead, all the results are for system-system experiments, with a view to later evaluation in a human-system setting.

One important advantage is that our planner is easy to use without expert knowledge of the underlying theory. In fact it is relatively easy to specify a dialogue problem without such knowledge by using the same hierarchical plan rules [Sacerdoti, 1974] that are commonly used to specify dialogue plans. The belief information is then acquired automatically from dialogue histories, so that the system will adapt itself over time. This is done using the planner's belief revision component, which updates beliefs in response to dialogue acts.

Our negotiation model is similar to the RMM of Gmytrasiewicz and Durfee [2001], in which game matrices are used as the nodes in a tree structure that branches on the beliefs of the agents. Our model differs in that we start with a given set of plan rules, from which a game tree is constructed. It is possible to then convert our game tree to normalised form, at which point we would obtain an RMM structure. However, the raw game tree is equally useful. The advantage of our planner is that it can work from the source plan rules, not requiring a game matrix to be provided. The negotiation acts are based on value of information in that an agent computes the change in utility of a plan as a result of obtaining a piece of information. Our set of negotiation acts is similar to those found in this paper.

Input to the planner is a nested belief model. Each level of the belief model has beliefs about the domain state, about the agent's capabilities in terms of hierarchical plan rules, and about plan recognition rules in terms of probability distributions over explanations for a given act in the plan structure. Figure 1 is an illustration of a typical plan library, with the

first level of nesting on the left-hand-side, and the second level of nesting on the right-hand-side. In this case we have two libraries since we have two agents with different expertise - one who is good at making main courses, and one who is good at making desserts. This pair is used to initialise levels one and two, levels three and four, and so on. Instead, we could have used only one library, with which to initialise every level, and allowed the system to adapt each level to the statistics in the collection of dialogue histories, by using a belief revision process. For example, if agent two were to often make-cake, the plan rule at level two that make-dessert decomposes to make-cake would attain a high probability of belief, as would the belief have-eggs. In this problem, the first agent must choose between cooking an omelette and cooking fish. The problem is that while the omelette is preferred, it requires eggs, which means that the second agent cannot make a cake for dessert. However, if the second agent has fruit, he can make a fruit salad instead. Therefore the first agent would like to find out whether the second agent has fruit before making his decision, and this is the source of a negotiation dialogue between the agents. This example is perhaps the smallest domain-level plan that can be constructed. In larger problems, agents would construct a deep game tree at of many moves, and this would form the basis of a long negotiation dialogue which deals with many different beliefs.

The planner constructs a domain-level game tree for this problem, illustrated in figure 2. To do this, the first agent chooses an action. Then the first agent performs the plan recognition process that he expects of the second agent. This uses the third level beliefs, since the second agent is working at level two and is trying to use beliefs about the first agent at level three to reconstruct the first act in the plan. In plan recognition, we use a simplifying assumption of "focussing", whereby an agent will always complete one branch of a plan before developing another one. Focussing is something that happens in human dialogue [Grosz and Sidner, 1986] for the very reason that it restricts the plan recognition hypotheses. Under this assumption, the plan recogniser need only search

among focussed candidates to explain the dialogue history. This means that multiple explanations can only occur when the plan tree is full, and a parent must be added to the plan tree to open up a sibling branch. For each act in the agent's repertoire, there is a probability distribution stored for a list of candidate parents. Having performed plan recognition at level three, the plan is expanded by adding an act to the next open node in the plan tree, using the level two plan rules. Suppose a third act were to be added to the plan. Then the agent at level two would expect the agent planning at level three to call the planner at level four to obtain a plan with two acts. In turn, the planner at level four calls the planner at level five. Each of the alternative plan decompositions available to an agent produces an alternative at a choice node in the game tree. If the plan recogniser must add a parent to a full subtree, or if there is a precondition to an act whose satisfaction depends on the belief state of the agent, a chance node is added to the game tree, preceding the choice node.

To illustrate, consider the omelette problem. The first agent decomposes make-dinner to make-main-course and then to either of make-omelette or make-fish, so we end up with two branches in the game tree at the root node (figure 2). Taking the make-fish branch, the second agent is expected to infer the make-main-course parent, and its parent, make-dinner, using the beliefs at level three. These are enough parents to make an open tree, and so make-dessert is attached. make-dessert can be decomposed to make-cake or to make-fruit-salad. Each of these has a precondition, and so a chance node is introduced to the game tree. Then, a choice node is added to each of the chance node outcomes.

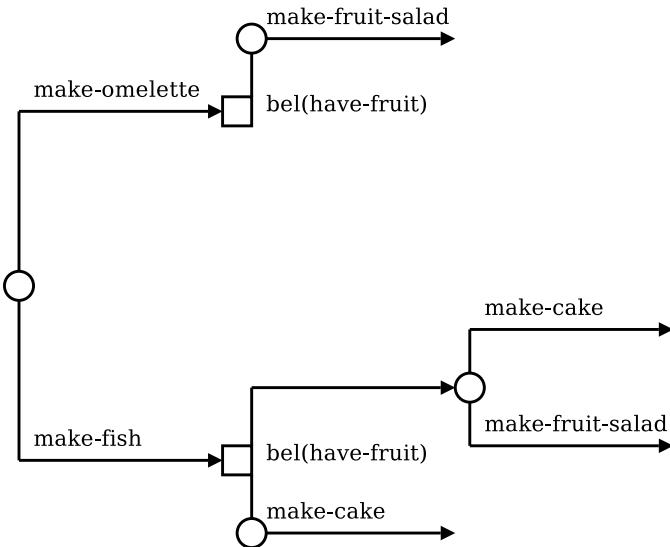


Figure 2: Game Tree

## 2.1 Evaluation of the game tree

Once constructed, the game tree must be evaluated in the context of a given belief state. Starting at the root of the tree, the first agent's nested belief state is taken, and the probabilities of the chance nodes are found, alternating between

level one and level two as the turn passes along the length of the dialogue. The value of a chance node is given as the weighted sum of its children, according to the expected utility rule. To evaluate a choice node, the remainder of the dialogue is evaluated from the point of view of the agent acting at the choice node. For example, the second choice node would be evaluated at level two, with its sequence of chance nodes being evaluated by alternating between levels two and level three. The agent then chooses the branch with the greatest value. Notice that evaluation of the plan tree requires a belief model that is only as deep as the number of steps in the plan. This allows infinite concepts like mutual belief to be represented within a finite model. At each step of evaluation, the planner's belief revision module is invoked on the belief model. For each dialogue act, preconditions and effects are added to every second level of the belief model, according to the precondition and effect rules contained in the beliefs at the level of the evaluating agent. This is a lazy but easy approach, since the revising agent only revises his beliefs about the other agent - he doesn't go so far as to update his private beliefs, and so does not have to resolve conflicts between each agent's beliefs. That is why every second level is updated, rather than every level. Neither does the agent make any further inferences or maintain the consistency of his belief set once revisions are made. This lazy approach has worked well for all of our example problems, but there are dialogues in which it would be useful to make deeper inferences, such as correction of misconception.

## 3 Negotiation acts

In addition to the domain level acts, which are supplied in the planner's input, the system has a set of built-in negotiation speech acts, which are used to exchange information before the domain level plan is executed. These acts result in a domain level plan that has a higher expected utility, through the agent revising his beliefs. Negotiation planning is a lot like domain-level planning, in that the negotiation acts are part of the game tree and go through the evaluation process in the same way. In fact the domain plan is attached at the leaves of the negotiation plan. However, only some of the negotiation acts can be specified using our plan rules and belief revision mechanism. Others require special more sophisticated belief revision operations to define them. Each is explained in turn.

### 3.1 "pass"

Pass is intended to allow an agent to pass the turn in a dialogue without saying anything. As such it has no preconditions and no effects. We have given pass a cost of 4, since in a dialogue there is usually some short utterance or a moment of silence before a pass can be inferred. It can be specified using the following plan rule:

name:	pass
parameter:	{ }
precondition:	{ }
effects:	{ }
decomposition:	{ }

### 3.2 "tell"

Tell is intended to allow an agent to inform another agent about a proposition. There is a form of tell for when the proposition is believed, and a form for when its negation is believed. tell-true has a precondition that the agent believes the proposition, and tell-false has a precondition that the agent believes its negation. Using the standard belief revision mechanism, the hearer performs a belief revision step in response to tell, adopting the belief that the precondition held for the speaker, and thus the belief is transferred. Tell has been given a uniform cost of 10.

```

name:          tell-true
parameter:     P
precondition:  bel(P)
effects:       {}
decomposition: {}

name:          tell-false
parameter:     P
precondition:  bel(not(P))
effects:       {}
decomposition: {}

```

### 3.3 "ask"

Ask has two pragmatic forms. ask-forced is representative of requests for information, for which the hearer's response is always to provide it without question. ask-forced has a cost of 10. It is represented by the following plan rules:

```

name:          ask-pair
parameter:     P
precondition:  {}
effects:       {}
decomposition: { ask-forced; reply(P) }

name:          reply
parameter:     P
precondition:  {}
effects:       {}
decomposition: { tell-true(P), tell-false(P) }

```

Ask provides another good illustration of the workings of the planner. The speaker plans an ask-pair, decomposes it, and produces the first act, ask-forced. This forms the single branch of the root node of the game tree. The hearer observes the ask-forced, and realising that the ask-forced node constitutes a full subtree, uses beliefs at level three to infer its parent, ask-pair, which can then be expanded, using level two beliefs, to a reply act. Then, reply is decomposed, giving two alternatives, tell-true and tell-false. Since each of these has a precondition which depends on the belief of the acting agent, a chance node is inserted into the game tree. In the true branch of this chance node, reply can be decomposed only to tell-true, and in the false branch, only to tell false. This ask game tree is shown in the upper branch of the tree in figure 3.

The second pragmatic form of ask is ask-auto, which allows the hearer to reply only if it is rational for him. An English example would be "I would like to know P". The

hearer is only required to revise his beliefs to accommodate the fact that the speaker "would like to know", and then decide to answer in the context of this revised model. Unfortunately, there is no simple way to revise the hearer's beliefs, since there could be many different reasons that motivate the speaker. For example, an agent might ask whether there is fruit because he believes that fruit salad is a good candidate plan, or because he prefers to paint a still life. Each explanation requires revision of different beliefs. To cope with this problem, we use a search algorithm, which searches the belief space for a state in which the speaker's asking is rational, but is at the same time as close as possible to the current belief state. The space is treated as a Euclidean space with a dimension for each belief in the belief model. While crude, this mechanism turns out to be effective, in that it produces worthwhile dialogues.

### 3.4 "propose"

Propose is used by agents to express their preferences over plans, such as in "I would choose P". Like ask-auto, the hearer responds by revising his beliefs, but there is the problem of many explanations that would cause the agent to prefer the plan. Once again, the agent chooses the explanation that has the smallest Euclidean distance from the current belief state. This definition of propose is more useful than that given by Gmytrasiewicz and Durfee [2001]. In their model, the agent merely prunes the game tree. In our model, beliefs are updated instead, with the side effect that the desired branch in the game tree is selected. Propose has a cost of 10.

### 3.5 Selection of the repertoire

We cannot claim that the set of acts produces the most efficient dialogues, but it seems unlikely that there would be any other acts as simple as these. The repertoire covers the obvious basic units of the agent's mental state, namely his beliefs and preferences, and covers all of the simple pragmatic definitions that are possible within the bounds of the belief revision mechanism. They also correspond well with acts seen in human collaborative planning, such as in the TRAINS [Allen, 1995] corpus, and those that appear in speech act theory and work on communication languages in artificial multi-agent systems. While perhaps not efficient, it is clear that they can eventually produce the most efficient domain plan, since using tell alone on every belief will lead each agent to a perfect model of the other.

One act that we haven't included is request, which is to propose as ask-forced is to ask-auto, in that it obligates the hearer to act. However, propose is already quite powerful in that the second agent searches the belief space from level three upwards to accommodate the proposed plan. For request to dominate propose, request must demand revisions at level two as well. However, this violates our assumption of lazy belief revision in which the second agent never revises his private beliefs. We suspect though that request would be useful with a less lazy belief revision mechanism.

## 4 Experiments

We have exercised the planner on a series of dialogue problems intended for use with a kitchen-assistant-robot, who

must coordinate his plan with the user so that they harmoniously prepare a meal together. These experiments are intended to show primarily that each of the negotiation acts in the agent's repertoire is a necessary member, in that it dominates all of the others in some examples. They also characterise some common decision problems seen in negotiation dialogues, and show how these decisions are subtly dependent on probabilistic values in the belief model, rather than on logical values.

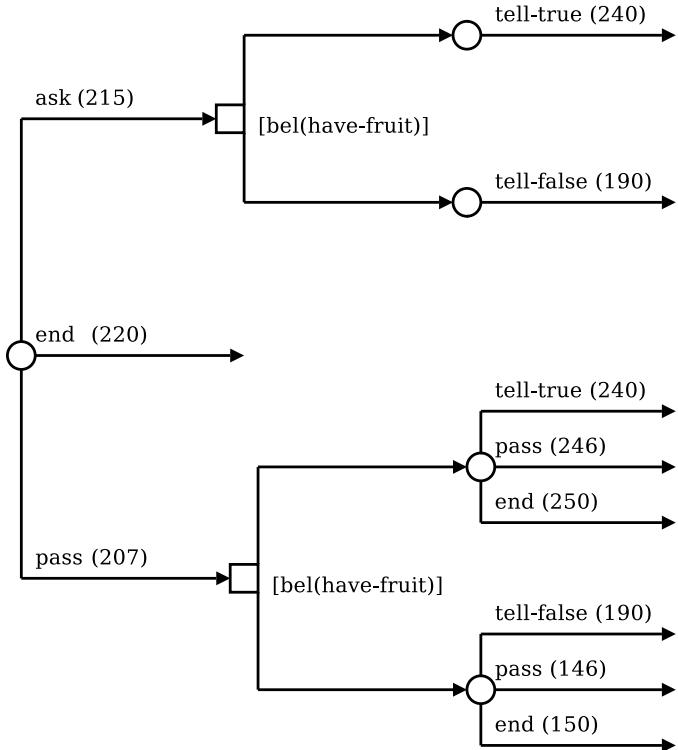


Figure 3: Game Tree in Experiment One

#### 4.1 Experiment One

Experiment 1 demonstrates the competition of initiative between asking a question and waiting to be told, in the context of the have-fruit problem introduced earlier. Asking is more expensive than waiting since it involves an ask act and a reply act, both of which cost 10 units. On the other hand, the agent can pass at a cost of 4 and risk that the second agent will decide to tell him the answer without being asked, at a cost of 10. At a total of 14 units this costs less, but it is risky. The planner generates the game tree in figure 3 for the dialogue, which for illustration is evaluated at the point [0.7,0.7] at levels [2,4] in a belief space which represents the belief have-fruit at levels 2, 4, 6, and so on.

The overall efficiency of each strategy is plotted in figure 4, with the level two belief along the left axis, and the level four belief along the right axis. Notice that the belief needs to be evaluated at both levels two and four to make the decision, and that all three strategies of asking, waiting and proceeding

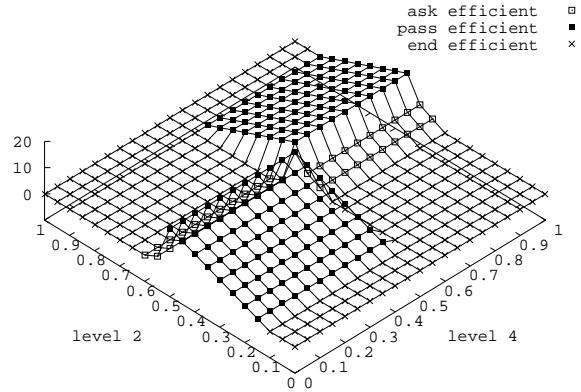


Figure 4: Utility of Strategies in Experiment One

directly to the domain-level plan are dominant in different regions of the belief space. It is clear from examining each strategy's curve that a probabilistic approach must be taken to modelling the agent's beliefs. There are sharp decision surfaces across which utility changes dramatically, and so an inaccurate belief model can make a significant difference to the performance of the system. Notice as well that there appears to be no simple rule, short of evaluating the game tree using both levels two and four, that would be satisfactory to solve this problem.

#### 4.2 Experiment Two

In experiment two we introduced a second variable which modifies the efficiency of waiting to be told. A precondition "have-eggs" is used with the make-omelette strategy, giving the game tree in figure 5. The result is that if the second agent believes there are no eggs, he will not bother to tell the first agent that he has fruit, even though the first agent privately believes that he has eggs. This demonstrates that the relative efficiency of asking and waiting to be told can vary. In fact, below a certain threshold, telling becomes always inefficient for the second agent. This is just a small example of the general phenomenon of competition for negotiation between different plans in a larger game tree. If a chance node appears near the root, the value of each of its branches with respect to each agent makes all the difference in deciding whether a plan attached to those branches is worth negotiating. As a result, agents try to either hold or decline the floor with respect to negotiation topics. In this instance, the first agent sees that his plan is more important than it is perceived, and aggressively takes the floor by asking.

#### 4.3 Experiment Three

Experiment three investigates the propose act, showing how it can be more efficient than an equivalent set of tell acts, or any other combination of acts. Propose is most useful when it is communicating a choice that is unexpected by the other agent, since by doing so, many beliefs can be significantly revised using just one act.

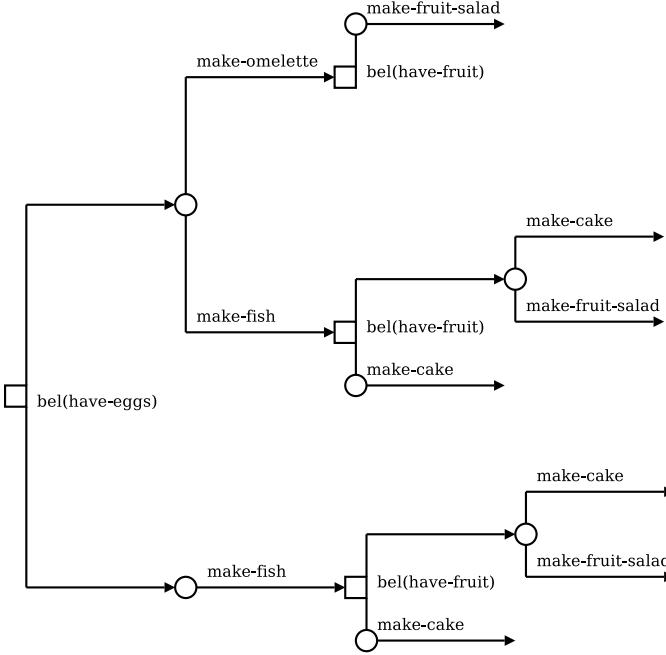


Figure 5: Game Tree for Experiment Two

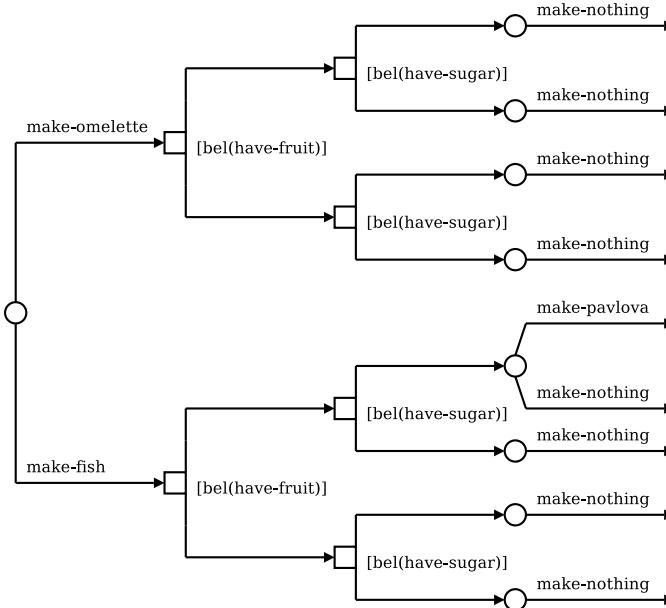


Figure 6: Game Tree for Experiment Three

We constructed a problem in which the second agent would prefer to make a pavlova (figure 6). Unfortunately, he will have to tell the first agent both that he has sugar, and that he has fruit if the first agent is to be convinced to choose make-fish, and leave the eggs for the pavlova. This is relatively expensive. The same effect can be achieved by simply proposing to make a pavlova. Then, the belief revision mechanism revises the hearer beliefs to a state in which both of the beliefs are high enough that the proposal is accommodated.

Negotiation acts were added to the agent's repertoire one by one to demonstrate the utility gain offered by each. To start, there were no negotiation acts, and so the game tree just consisted of the domain-level tree, with a value of 100 for make-omelette. Next, the pass and tell acts were added. This produced the negotiation game tree in the upper part of figure 8. This tree shows the best strategy only for the agent, so that each choice node is pruned down to only one alternative. Notice that in response to a pass, the second agent uses a pair of informs in the **true,true** branch of the game tree. This sub-dialogue is efficient, and since it happens in one quarter of instances, the value for the tree is 102.5, which is a marginal gain over the 100 obtained from the plain domain-level plan. Next, the ask acts were added, but these were dominated by the pass and inform combination, and so the same result of 102.5 was obtained. Next, propose was added. This produced the tree in the bottom part of figure 8, with propose dominating instead of tell. Since the negotiation ends with the proposal, there is a smaller cost than in the upper game tree in figure 8. The overall cost of the dialogue turns out to be 106, compared with 102.5 obtained without using propose.

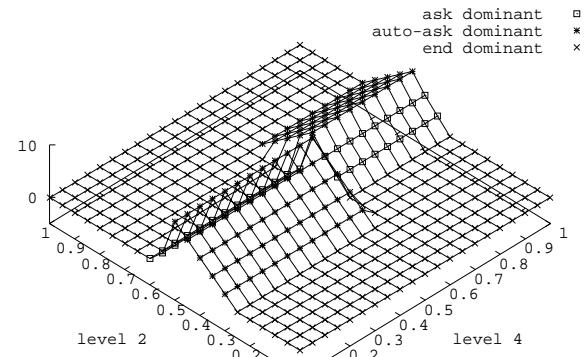


Figure 7: Dominance of ask-auto over ask-forced

#### 4.4 Experiment Four

In experiment four, we compared the two pragmatic senses of ask, to demonstrate that both are required for efficient dialogue. We took the problem that was used in experiment two, where it happened that waiting to be told was never efficient, but asking was. Ask-auto causes the second agent to revise his beliefs about the eggs in the search for a belief state in which asking would be efficient for the first agent. After that, the agent uses a tell, which is now efficient because of the eggs revision. ask-auto therefore has a similar character to waiting for a tell, but is a little less efficient due to the cost of having to ask. The reason that ask-auto has an advantage over ask-forced is that ask-auto uses the level 4 belief in making the tell decision, whereas ask-forced only uses the level 2 belief. Figure 7 shows that both ask-forced and ask-auto are dominant at different times, whereas waiting for a tell is

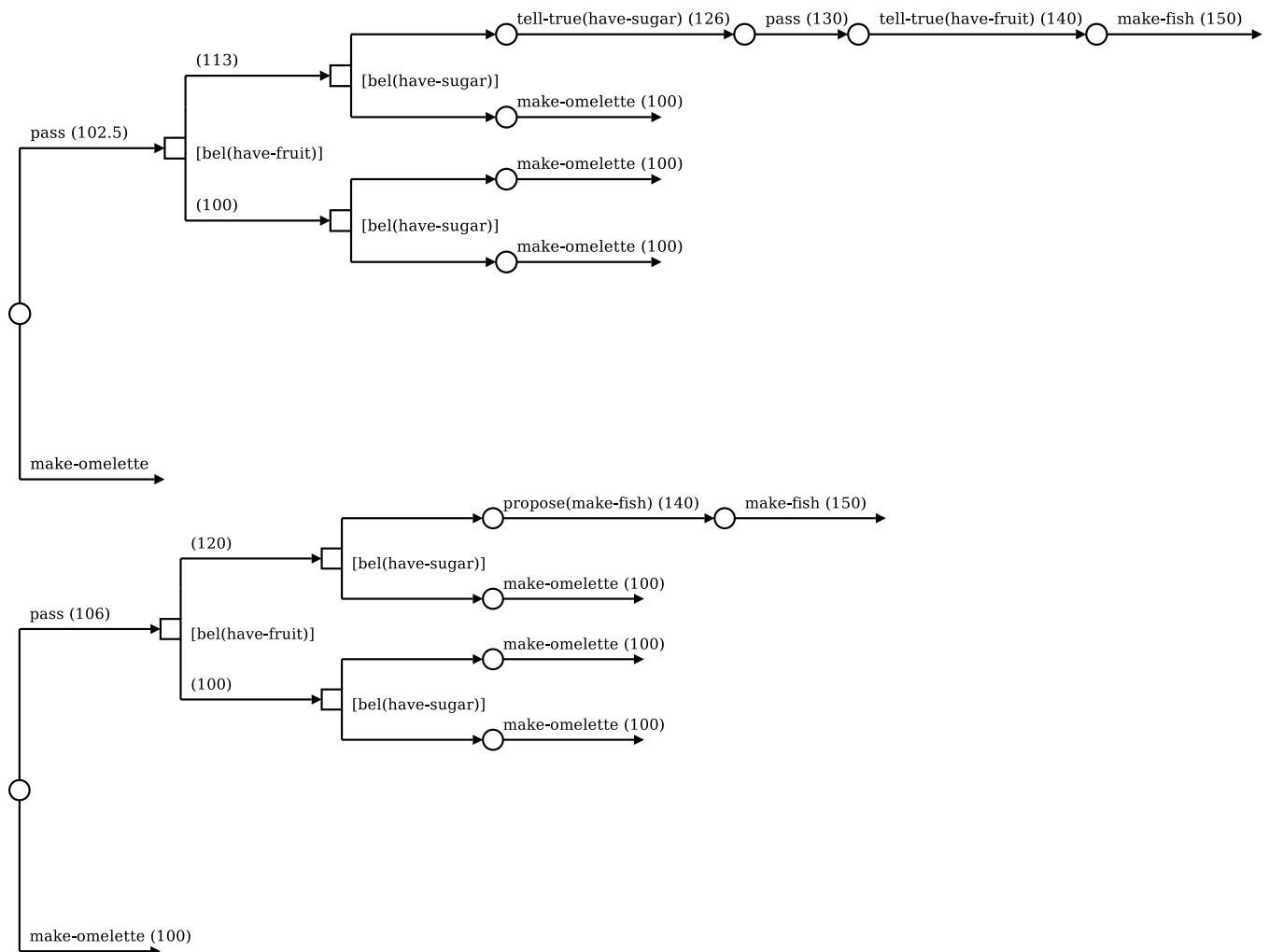


Figure 8: Negotiation Game Trees for Experiment 3

never dominant. Therefore each sense of asking is included in the repertoire.

## 5 Conclusion

We have described a planner that uses game trees and a probabilistic user model to produce adaptive dialogue strategies. We described a repertoire of negotiation acts - pass, ask, tell, and propose, each of which complements the others. The use of a probabilistic model was shown to be important to achieving efficient dialogue, since the utility of the strategies varies across a probabilistic belief space.

## References

- [Allen and Perrault, 1980] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.
- [Allen, 1995] James F. Allen. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI (JETAI)*, 7:7–48, 1995.
- [Carberry, 1990] Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, 1990.
- [Carletta, 1992] Jean Carletta. Planning to fail, not failing to plan: risk-taking and recovery in task-oriented dialogue. In *Proceedings of the 14th conference on Computational linguistics*, pages 896–900. Association for Computational Linguistics, 1992.
- [Gmytrasiewicz and Durfee, 2001] Piotr J. Gmytrasiewicz and Edmund H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 4(3):233–272, 2001.
- [Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguit.*, 12(3):175–204, 1986.
- [Heeman and Hirst, 1995] Peter A. Heeman and Graeme Hirst. Collaborating on referring expressions. *Comput. Linguit.*, 21(3):351–382, 1995.
- [McCoy, 1989] K. F. McCoy. Highlighting a user model to respond to misconceptions. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 233–254. Springer, Berlin, Heidelberg, 1989.
- [McEleney and O’Hare, 2005] Bryan McEleney and Gregory O’Hare. Decision theoretic planning for initiative problems. In *Proceedings of the 10th International Conference on User Modelling*, Edinburgh, Scotland, September 2005. Springer Verlag.
- [Pollack, 1986] Martha E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. In *Proceedings of the 24th conference on Association for Computational Linguistics*, pages 207–214. Association for Computational Linguistics, 1986.
- [Sacerdoti, 1974] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

# An Algorithm that Continuously Seeks Minimum Length Dialogs

R. Bryce Inouye and Alan W. Biermann

Duke University

Department of Computer Science

{rbi, awb}@cs.duke.edu

## Abstract

We demonstrate via simulation the efficacy of a methodology for continuously updating a dialog strategy to seek the minimum expected length dialog for a dialog system. The algorithm requires that the system be able to calculate efficiently the optimal dialog strategy given a model of the probability that the system's users will be able to communicate various pieces of information. We present an example using fixed initiative dialogs, for which efficient optimization methods are available. Expected dialog lengths are calculated from a user model which stores the system's estimation of the probability that the user has the requested knowledge. The system can adapt its behaviors to account for drastic changes in the user responses.

## 1 User Modeling and the Space of all Possible Dialog Strategies

We examine the case where a dialog system interacts with a population of users in an unending sequence of dialogs. The system will have the ability to modify its behavior over time, and its basis for change will be the responses from users. We model the response from a user as either successful or not and compute the probability of success as an ongoing and changing value. If the user population is stable in its response probabilities, the system will slowly converge on the optimum dialog strategy. If the user population varies in its response probabilities, the system will track those changes and seek new optimums. Our measure of dialog quality is the length, in number of turns, of the dialog. We value short dialogs above long ones following the tradition of many researchers.

Our approach involves high performance techniques for finding the optimum dialog strategy among the set of all possible such strategies that are being considered. The techniques assume that the system has estimates of the probabilities of successful answers from the user for each question type. Initially, our algorithm will guess at the response probabilities, select the optimum strategy from the space of all strategies, and implement that strategy until additional gathered data on probabilities indicates some change is needed. We have found substantial subclasses for which this computa-

tion can be done in polynomial time, and for those subclasses, we have a complete solution to the dialog learning problem.

The dialog subclasses that we will examine here are fixed-initiative dialogs with certain additional properties that we will present in this paper. We will study only dialogs where the machine leads the interaction and where the user is asked to answer or fail to answer the requests. We make additional restrictions on these subclasses as will be explained below. We are not discouraged with the seemingly limited applicability of these results since fixed-initiative dialogs are quite common. We also note that under our definition of dialog initiative, even mixed-initiative dialogs are made up of sequences and hierarchies of fixed-initiative segments so we can often optimize those segments and make a major step towards optimization of general, mixed-initiative dialogs.

## 2 The Adaptive Algorithm

Our model assumes that there is a finite number of utterances (or utterance types) that the machine can output. Associated with each output there is the machine's estimate  $p$  of the probability that the user can respond successfully. For example, the machine might request that the user give an identification number. The user might successfully provide it and  $p$  will give the machine's estimate of the probability that this will occur. However, the user might not know the identification number or speech recognition errors might cause failure of the system to obtain it. These would be classed as unsuccessful responses to the query. The collection of all such probabilities  $p$  will be denoted here as the user model  $\mathcal{P}$ . Initially,  $\mathcal{P}$  might be filled with 0.5 values or some estimates selected by the designer. After many dialogs, the system should be able to gather enough data on the probabilities of successful responses to make improved estimates.

In its most basic form, our algorithm is as given here.

Algorithm for seeking minimum average length dialogs.

Initialize user response probabilities  $\mathcal{P}$ .

Repeat

1. Select strategy  $S$  which has minimum expected dialog length for the given  $\mathcal{P}$ .
2. Execute current dialog using  $S$ .
3. If recent user responses indicate that  $\mathcal{P}$  is not current, update  $\mathcal{P}$ .

### 3 A Class of Solved Problems

We present in [Inouye and Biermann, 2005] a model for fixed-initiative dialogs and solutions to a number of special cases. The model asserts that the machine has the problem of completing a task and its only behavior is one of asking questions  $q_i$  and receiving answers associated with them. It assumes that there are rule sets  $r = \{q_1, q_2, q_3, \dots, q_n\}$  of such questions such that if all the questions in  $r$  are satisfactorily answered by the human user, the task will be successfully completed. If any question of  $r$  fails, then  $r$  will not successfully achieve the task. The model also assumes that there may be several or even many such rule sets,  $r_1, r_2, \dots, r_m$  and the satisfactory completion of any one of these will be sufficient to solve the overall task. This model is derived from planning-based approaches to dialog [Litman and Allen, 1987] or theorem-proving approaches [Smith *et al.*, 1995] where it can be shown that for any problem statement there must exist finite sets of questions which, if all answered successfully, will yield a successful solution to the top-level goal. The latter approach follows the theory of dialog presented by Grosz and Sidner [1986] where the theorem proving machinery implements the “intentional structure” that they describe.

A strategy for achieving the task is to find an ordering of the questions in the rule sets  $r_1, r_2, \dots, r_m$  such that the task will be solved if there is a solution. An optimal strategy will be one which orders those questions such that the expected length of the dialogs will be minimum given the probabilities that are associated with the questions.

[Inouye and Biermann, 2005] shows solutions to several versions of the problem. Stated in very concise terms, some of the results are:

1. When there is only one rule set  $r$ : Here the optimum order for asking questions is to ask the questions in order of increasing probability of success. Thus the question of least probability of success is asked first.
2. When there are many rule sets, each containing only one question: The optimum order for asking questions is the question with highest probability for success first and then, in order, to least probable question last.
3. When there are many rule sets with the property that no two rule sets have a question in common: The technique is to compute for each rule set  $r_i$  the expected number  $f_i$  of questions asked if the rule set fails and the probability  $s_i$  that the rule set will succeed. The optimum strategy computes the value for each rule set  $r_i$  the quantity  $v_i = n_i + f_i(\frac{1}{s_i} - 1)$  and will ask questions from the rule sets in the order of  $v_i$  from least to greatest. The questions within each  $r_i$  must be asked in the order of increasing probability of success.
4. In the general fixed-initiative case: An exact solution can be computed by generating all possible dialog strategies and evaluating the expected length of dialog for each, then selecting the strategy that yields the minimum value. However, the number of possible strategies is exponential in the number of rule sets and the number of questions, making computation of the optimal solution intractable in most cases.

We give a heuristic solution for the general case and show its effectiveness in lowering expected dialog lengths. The method is to order the rule sets  $r$  according to rule 3 above, even if some rule sets do have questions in common. Within each rule set, order the questions from beginning to end with least probable questions first.

### 4 An Example

Next we demonstrate our algorithm on a realistic example. Suppose a telephone company has an automated system to handle repair requests. We assume that the strategy is to ask a series of questions that identify the telephone with the problem and the time and specific complaint about the phone. We also assume that some phones will not work well enough to report the problem or may fail during the call. So there is a second series of questions to identify the caller and any alternative phone available to the caller where the phone company may contact that person. This yields the model  $r_1 = \{q_{11}, q_{12}, q_{13}, q_{14}, q_{15}\}$ ,  $r_2 = \{q_{21}, q_{22}, q_{23}\}$  where we assume the questions might be as follows:

$q_{11}$  = “Please give your telephone number.”

$q_{12}$  = “Please give the area code of your phone.”

$q_{13}$  = “Please give your billing address.”

$q_{14}$  = “When did your problem first occur?”

$q_{15}$  = “Describe the problem you have encountered.”

$q_{21}$  = “Can you give me your billing account code for your telephone?”

$q_{22}$  = “What is your last name?”

$q_{23}$  = “What number should we call to discuss your problem?”

We have simulated the results of several hundred interactions between the system and a group of users whose knowledge bases can be characterized by the probability that any user might possess a particular piece of knowledge. Figure 1 illustrates the basic architecture of the simulator. The box in the upper left of Figure 1 represents the dialog system. It consists of a knowledge base containing the dialog rule sets that tell how to accomplish the dialog task, a user model that stores  $\mathcal{P}$ , which is the system’s estimates of the probabilities that an arbitrary user will be able to answer a given question successfully, and an optimization routine that takes the user model and the dialog rule sets and produces the optimal dialog strategy with respect to the user model as described in Section 3. A dialog strategy specifies completely which question the system should ask the user at any point in the dialog.

The box in the upper right of Figure 1 represents the simulated user generator. It consists of a random number generator and a knowledge distribution function  $k$  that takes as input a question and returns the actual probability that a user will be able to successfully answer the question. Note that the user model is system’s estimate of the knowledge distribution function. The simulated user generator creates simulated users by producing a knowledge base for the user. It uses the random number generator and the knowledge distribution function to determine for each possible question if the user will successfully answer that question.

To simulate a dialog, the simulator takes a dialog strategy provided by the system and a simulated user. The dialog strategy (which can be represented as a tree) indicates which question should be asked first. The simulator checks to see if the user can answer the question successfully, and the result is reported to the system. The dialog strategy is consulted again to see what question should be asked based on the previous response, and the process repeats until the dialog terminates in either success or failure.

The simulator employs the following loop as its basic control structure:

1. Calculate the optimal dialog strategy with respect to the current user model.
2. Check to see if the knowledge distribution function  $k$  has changed.
3. Create a new simulated user according to  $k$ .
4. Calculate the results of a dialog between the system following the optimal strategy and the simulated user.
5. Use the results of the dialog to update the user model. Specifically, for each question  $q_i$  revise its associated estimated probability  $p_i$  of success as follows:  $p_i = (\text{number of times } q_i \text{ has been successfully answered in recent history}) / (\text{number of times } q_i \text{ has been asked in recent history})$
6. Repeat.

Note that once the simulated user and the dialog strategy have been generated, the resulting dialog is deterministic.

If the user model and the knowledge distribution function are the same, then the system is guaranteed to produce the optimal dialog strategy. However, in general, there is some error in the user model. We have chosen to characterize the error using a root mean square difference:

$$\text{error} = \sqrt{\frac{1}{n} \sum_{\text{all } q} (k(q) - p_q)^2}.$$

As the error in the user model decreases, the quality of the computed optimal solution increases. This computation enables us to observe the improvement of the user model during a simulation. However, our computed error does not affect the operation of the dialog system or the simulation.

The simulator then takes the results of the interaction, as observed by the system, and updates the system's user model. The updated user model is then used to recalculate a new optimal strategy for the system, and the process repeats itself. (We do not know what the best plan is for how often to recompute the optimal strategy. In our simulation, it was easy to recompute the optimum after every simulated dialog. But in other situations, the system might stay with the currently selected strategy until a criterion is met to seek a new optimum.)

We present the results of one run of our simulation. Our simulator implements the algorithms described in section 3 in its dialog strategy optimizer. The dialog scenario consists of two rule sets,  $r_1 = \{q_{11}, q_{12}, q_{13}, q_{14}, q_{15}\}$  and  $r_2 = \{q_{21}, q_{22}, q_{23}\}$ . The initial user model for the system assigns

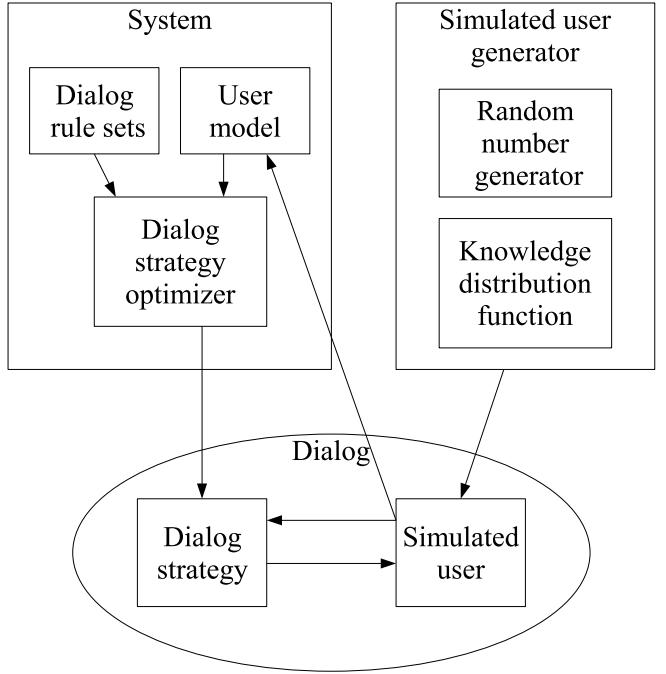


Figure 1: The dialog simulator.

a value of 0.5 to each question. The initial knowledge distribution function  $k$ , which characterizes user responses but is not known to the system, has the value 0.2 for  $q_{21}$  and 0.98 for all other questions.

If this problem were handled by conventional means, all eight questions would be asked for each dialog. We show in this example what improvements are possible with an adaptive system. In fact, the average number of questions asked changes during the simulation and ends after several hundred dialogs just above three questions per dialog. Specifically, our system asks questions from one of  $r_1$  or  $r_2$  until it succeeds or fails. If it succeeds, the dialog halts and if it fails, the system proceeds to the other rule set,  $r_1$  or  $r_2$ . With our automatic system selecting which rule set to choose and what order of the questions should be asked within a rule set, substantial additional improvements can be achieved. The graph of Figure 2 shows the results.

The initial optimum version of the dialog based upon the first probability estimates has the system asking questions from  $r_2$  initially. The initial average length of dialogs was about 5.2 questions as shown by point S1 in Figure 2.

After nine dialogs, the system has made enough observations to realize its initial guess at probabilities (all are 0.5) are incorrect and it changes strategy (point C1). The system begins asking questions from  $r_1$  first. The root-mean-square of the distance between the internally stored probabilities and the actual probabilities being used in the simulation is given by the root-mean-square curve.

After 100 dialogs, we assume that a new telephone problem has arisen which callers cannot successfully explain to our speech recognition system. Suddenly the actual probability of successful answers to  $q_{15}$  falls from 0.98 to 0.4. The

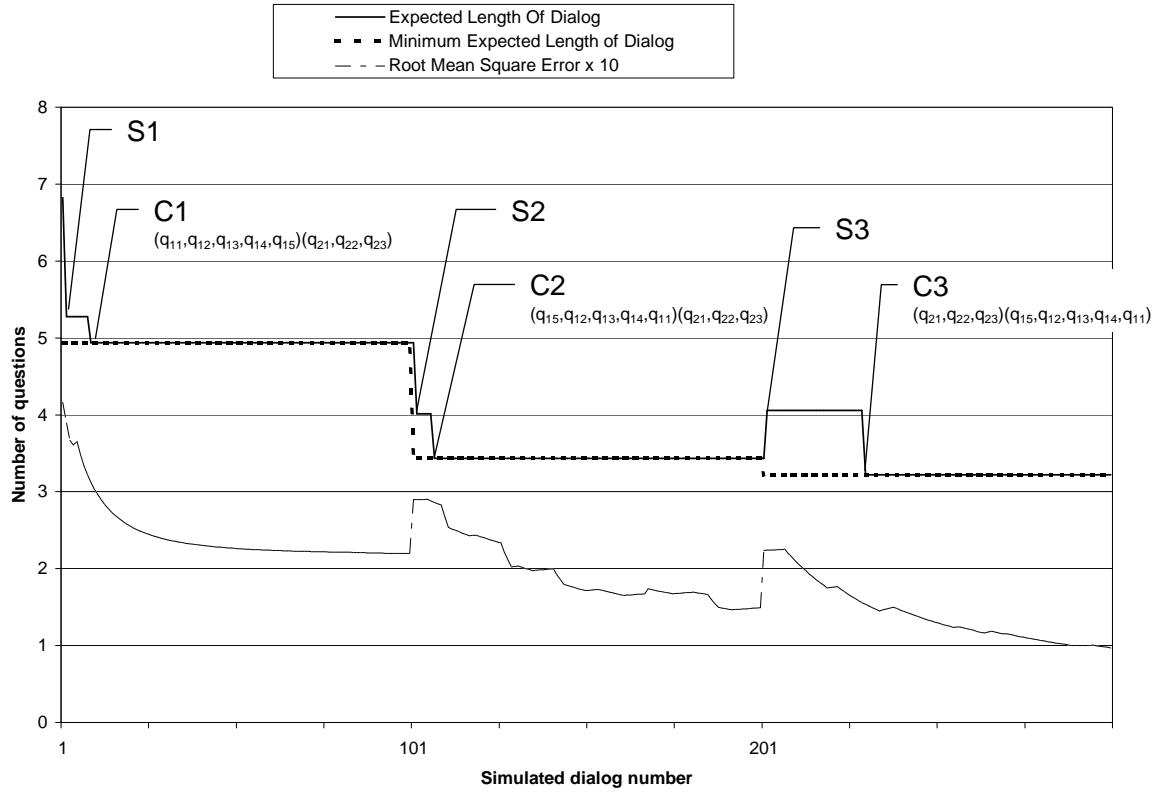


Figure 2: Results of a simulation. The solid line traces the expected dialog length given the knowledge distribution function  $k$  and the system's dialog strategy at the time. The heavy dotted line represents the expected value of the dialog length for the optimal dialog strategy given  $k$ . The lower dotted line traces the value of  $\sqrt{\sum_{allq} (k(q) - p_q)^2}$ , the RMS error in the user model, multiplied by 10 to emphasize the scale of the changes. The optimal dialog strategy learned by the system is indicated at points C1, C2, and C3. Rule sets are grouped by parentheses; the system asks questions in the order in which they appear, but if a question fails, causing a rule set to fail, the sequence skips the rest of the questions in the rule set and continues at the beginning of the next rule set, if one is available.

system encounters a change in user success probabilities at point S2 and has enough updated success statistics at point C2 to alter its strategy again. Here it moves  $q_{15}$  to the front of its  $r_1$  series of questions. The average length of dialogs has now dropped to 3.4 questions. The average root mean square distance from the internal probabilities and the actual external ones has jumped up again. The system gathers data and slowly decreases this disparity.

Then we assume, after dialog 200, that account codes are suddenly much easier for callers to access (point S3). (Perhaps they are printed in red on the new bills.) The probability for  $q_{21}$  increases from 0.2 to 0.7. The expected length of the system's dialogues jumps to over 4 questions per dialog. However, the system adapts again (point C3) moving  $r_2$  back to the first set and giving an expected dialog length of 3.2 questions per dialog.

The simulation shows the system under a variety of realistic situations and demonstrates how it can change its strategy to minimize expected dialog length. In the workshop presentation, we will give a series of much more ambitious examples and show the power of our algorithm in these cases.

## 5 Discussion

Others have built learning systems for dialog. One example is the NJFun system [Singh *et al.*, 2002; Litman *et al.*, 2000]. In this system, the authors designed an MDP state representation for their dialog environment and the system learned the preferred transitions in the state diagram that were correlated with successful interactions. This system substantially improved its dialog success rate after a series of subjects used it. Our system solves a smaller class of problems than the NJFun group since they allow their designers to do detailed specifications of the states and the possible transitions. We solve only part of the narrow fixed-initiative class, but our system completely self-organizes for optimal behavior, as measured by dialog length, during a series of dialogs.

The POMDP is an attractive model since speech recognition can lead to serious errors in perception of the state of the dialog. This model has been adopted by Roy, Pineau, and Thrun [2000] and it can be used to achieve learning. Their approach was to represent user intention as the state of the system and speech recognition was used to partially observe this state. In an application to nursing home robot control, their system achieved significant improved performance through learning. A difficulty with this approach comes from the high cost of finding optimum POMDP solutions and approximation schemes are usually part of the methodology. Again, our system does not require as much design effort, but it solves a narrower class of problems optimally.

Our use of simulated dialog system users to evaluate the performance of our algorithm follows the work of others who have used simulation techniques to avoid the high costs of obtaining actual human-computer dialogs. Guinn [1996] relies upon simulated users in his study of the effects of varying the initiative taking strategies of a dialog system, and uses the technique of assigning knowledge according to a supplied probability distribution. Chung [2004] employs simulated users to aid system development and train speech

recognition and understanding components, and uses probabilities to characterize not only the knowledge, but the behavior of the simulated user, as the system is designed to support mixed-initiative interactions. Scheffler and Young [2001] simulate users using a similar data structure, consisting of an attribute-value structure that keeps track of the fields that must be instantiated to achieve a goal. Another approach to simulated user generation [Levin *et al.*, 2000; Levin and Pieraccini, 1997; Levin *et al.*, 1998] does not model knowledge explicitly, but instead uses utterance-level bigrams to determine what responses the user will make.

## 6 Next Steps

The dialog model we use in this paper supports calculation of optimal dialog strategies for dialog scenarios with more complexity than the simple example presented in this paper. We are studying dialogs which are substantially more complicated than the one described here. Also, the current model considers only the set of fixed initiative dialogs. We define dialog initiative in such a way as to allow us to decompose mixed initiative dialogs into a sequence or hierarchy of fixed initiative dialogs, which can be optimized individually, and then combined to produce an optimized mixed initiative dialog strategy.

Another possible line of inquiry involves learning to decide when to recalculate a dialog strategy. If the system has a good user model, the dialog strategy will not change very often, and thus does not have to be recomputed after each dialog. Our example demonstrates that our method is capable of adapting to changes in the user environment simply by tracking changes in the user model. The method we use in the experiments reported in this paper can detect changes that occur over time. We would like to be able to detect sudden shifts in the user environment, corresponding to the occurrence of some external event, that would cause the system to have to change strategy suddenly.

## Acknowledgments

The authors gratefully acknowledge the ongoing financial support by SAIC of our work on spoken language systems. This project has also been supported by National Science Foundation Grant 0121211 and Defense Advanced research Projects Agency Grant N66001-01-1-8941.

## References

- [Chung, 2004] Grace Chung. Developing a flexible spoken dialog system using simulation. In *Proceedings of ACL '04*, Barcelona, Spain, July 2004.
- [Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.
- [Guinn, 1996] Curry I. Guinn. Mechanisms for mixed-initiative human-computer collaborative discourse. In *Proceedings of ACL96*, pages 278–285, Santa Cruz, CA, 1996.
- [Inouye and Biermann, 2005] R. Bryce Inouye and Alan W. Biermann. Minimizing the length of fixed initiative dialogs. *submitted for publication*, 2005.

- [Levin and Pieraccini, 1997] Esther Levin and Roberto Pieraccini. A stochastic model for learning dialogue strategies. In *Proceedings of the European Conference on Speech Communication and Technology, EUROSPEECH97.*, 1997.
- [Levin *et al.*, 1998] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using markov decision process for learning dialogue strategies. In *Proc. ICASSP*, 1998.
- [Levin *et al.*, 2000] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, January 2000.
- [Litman and Allen, 1987] Diane J. Litman and James F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200, 1987.
- [Litman *et al.*, 2000] Diane Litman, Michael S. Kearns, Satinder B.Singh, and Marilyn A.Walker. Automatic optimization of dialogue management. In *Proc. of COLING 2000*, 2000.
- [Roy *et al.*, 2000] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialog management using probabilistic reasoning. In *The Proceedings of the Association for Computational Linguistics*, 2000.
- [Scheffler and Young, 2001] K. Scheffler and S. Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation, 2001.
- [Singh *et al.*, 2002] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.
- [Smith *et al.*, 1995] Ronnie W. Smith, D. Richard Hipp, and Alan W. Biermann. An architecture for voice dialog systems based on prolog-style theorem proving. *Computational Linguistics*, 21(3):281–320, 1995.

# Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data

James Henderson and Oliver Lemon and Kallirroi Georgila

School of Informatics

University of Edinburgh

2 Buccleuch Place, Edinburgh EH8 9LW, United Kingdom

james.henderson@ed.ac.uk

## Abstract

We propose a method for learning dialogue management policies from a fixed dataset. The method is designed for use with “Information State Update” (ISU)-based dialogue systems, which represent the state of a dialogue as a large set of features, resulting in a very large state space and a very large policy space. To address the problem that any fixed dataset will only provide information about small portions of these state and policy spaces, we propose a hybrid model which combines reinforcement learning (RL) with supervised learning. The reinforcement learning is used to optimise a measure of dialogue reward, while the supervised learning is used to restrict the learnt policy to the portion of the space for which we have data. Linear function approximation is used to handle the large state space efficiently. We trained this model on a subset of the COMMUNICATOR corpus, to which we have added annotations for user actions and Information States. When tested with a user simulation trained on the same data, our model outperforms all the systems in the COMMUNICATOR data (it scores 37% higher than the best COMMUNICATOR system). All of these advances will improve techniques for bootstrapping and automatic optimisation of dialogue management policies from limited initial datasets.

## 1 Introduction

We investigate using a fixed corpus of dialogues to automatically optimise dialogue systems which have rich representations of dialogue context. The “Information State Update” (ISU) approach to dialogue [Larsson and Traum, 2000] employs such representations of dialogue context for flexible dialogue management, and the question arises as to whether dialogue management policies can be learnt [Levin and Pieraccini, 1997] for such systems. We focus on learning with a fixed corpus of dialogues because dialogue corpora are very expensive to produce, and it is often not practical to produce new dialogues during the course of learning. Even if dialogues can be automatically generated with simulated users,

training on simulated dialogues does not replace the need to fully exploit the real data.

Previous work on learning dialogue management policies has focused on small state spaces and small sets of actions to choose between [Singh *et al.*, 2002; Levin *et al.*, 2000; Scheffler and Young, 2002]. They use reinforcement learning (RL) to find a policy which optimises a reward function over dialogues. In this paper we address the ambitious task of learning to choose between a relatively large number of actions (70 in our experiments), with a very large state space (over  $10^{87}$  states are theoretically possible), given a fairly small corpus of dialogues (697 in our experiments). We use linear function approximation to handle the large state space, but this does not address the difficulty of searching for an optimal policy in the huge space of possible policies.

Given the huge policy space, any fixed dataset will only provide information about a small portion of this space. To address this problem, we propose a hybrid learning model which combines reinforcement learning (RL) with supervised learning. RL is used to optimise a measure of dialogue reward, while supervised learning is used to restrict the learnt policy to the portion of the space for which we have data. When trained on a subset of the COMMUNICATOR corpus [Walker *et al.*, 2001a; 2002] and tested with a user simulation trained on the same data, this model outperforms all the systems in the COMMUNICATOR data. When the relative importance of the RL component and the supervised learning component are adjusted, we currently find that a purely supervised model performs the best. However, for a range of degrees of influence of RL, the hybrid system still performs better than the COMMUNICATOR systems. In this paper, we first discuss the annotations we have added to the COMMUNICATOR data, then present the proposed learning method, and then present our evaluation method and the results.

## 2 Automatic Annotation of the COMMUNICATOR Data

The COMMUNICATOR corpora (2000 [Walker *et al.*, 2001a] and 2001 [Walker *et al.*, 2002]) consist of human-machine dialogues (approx 2300 dialogues in total). The users always try to book a flight, but they may also try to select a hotel or car-rental. The dialogues are primarily “slot-filling” dialogues, with some information being presented to the user af-

ter the system thinks it has filled the relevant slots. These corpora have been previously annotated using the DATE scheme, for each *system* utterance's Conversational Domain, Speech Act, and Task [Walker *et al.*, 2001b].

We used a hand-crafted automatic system to assign Speech Acts and Tasks to the user utterances, and to compute information states for each point in the dialogue (i.e. after every utterance). The system is implemented using DIPPER [Bos *et al.*, 2003] and OAA [Cheyer and Martin, 2001], using several OAA agents (see [Georgila *et al.*, 2005] for more details). An example of some of the types of information recorded in an information state is shown in figure 1. The state is intended to record all the information about the preceding portion of the dialogue which is relevant to making dialogue management decisions, including filled slots, confirmed (i.e. grounded) slots, and previous speech acts.

For the experiments reported in this paper, we used a preliminary version of the annotation, which included 4 of the 8 systems in the 2001 corpus. This subset consists of 97 users, 697 dialogues, and 51,309 total states. The fact that the annotation was done automatically means that some errors are inevitable, particularly in this preliminary version. But we believe that this has an equal effect on our performance measures for both the COMMUNICATOR systems and our learnt systems, so it does not affect our conclusions.

### 3 Using the Data for Reinforcement Learning

We use the annotated COMMUNICATOR data to train a Reinforcement Learning system. In RL, the objective of the system is to maximise the reward it gets during the course of the dialogue. Rewards are defined to reflect how well a dialogue went, so by maximising the total reward the system optimises the quality of dialogues. The difficulty is that, at any point in the dialogue, the system cannot be sure what will happen in the remainder of the dialogue, and thus cannot be sure what effect its actions will have on the total reward at the end of the dialogue. Thus the system must choose an action based on the average reward it has observed before when it has performed that action in states similar to the current one. This average is the expected future reward.

The core component of any RL system is the estimation of the expected future reward (the Q-function). Given a state and an action that could be taken in that state,<sup>1</sup> the Q-function tells us what total reward, on average, we can expect between taking that action and the end of the dialogue. Once we have this function, the optimal dialogue management policy reduces to simply choosing the action which maximises the expected future reward for the current state.

The actions which the reinforcement learning system needs to choose between are defined in terms of the DATE scheme [Walker and Passonneau, 2001] system annotations for Conversational Domain, Speech Act and Task. Each possible triple of values for these three features is considered a different action. In addition, there are release\_turn and

<sup>1</sup>The expected future reward also depends on the dialogue management policy which the system will use in the future. This self-referential nature of RL is the topic of much RL research, and will be discussed more below.

end\_dialogue actions. There are a total of 70 actions which occur in the data.

#### 3.1 Defining Dialogue Reward

To apply RL to the COMMUNICATOR data, we first have to define a mapping  $r(d, i)$  from a dialogue  $d$  and a position in that dialogue  $i$  to a reward value. This reward function is computed using the reward level of annotation in the COMMUNICATOR data, which was extracted from user questionnaires and task completion measures. For all states other than the final state, we provide a reward of -1. This encodes the idea that, all other things being equal, short dialogues are better than long ones. For the final state we provide a reward which is the sum of the rewards for each feature in the reward annotation. “Actual Task Completion” and “Perceived Task Completion” are both worth a reward of 100 if they are non-zero, and 0 otherwise. The remaining reward features have values ranging from 1 to 5 in the annotation. Their reward is their value (minus one) times the weight shown in table 1. The relative values of these later weights was determined by the empirical analysis reported in [Walker *et al.*, 2001a].

Actual Task Completion	100
Perceived Task Completion	100
Task Ease	9
Comprehension Ease	7
System behaved as Expected	8
Future Use	9

Table 1: The weights used to compute a dialogue’s final reward value. The first two features’ weights are multiplied by 0 or 1, and the rest are multiplied by values from 0 to 4.

#### 3.2 Estimating the Expected Future Reward

Given this definition of reward, we want to find an estimate  $Q(s_i, a)$  of the expected future reward, which is the expected value (“ $E[ ]$ ”) of the total reward between taking action  $a$  in state  $s_i$  until the end of the dialogue.

$$Q(s_i, a) \approx E[\sum_{j>i} r(d, j) | s_i, a]$$

Given that the number of possible future state sequences ( $s_{i+1}, \dots$ ) is exponential in the length of the sequences, it is not surprising that estimating the expected reward over these sequences can be very difficult.

The ISU framework is significantly different from the frameworks used in previous work on reinforcement learning for dialogue management, in that the number of possible states is extremely large. Having a large number of states is a more realistic scenario for a practical, flexible, and generic dialogue systems, but it also makes many RL approaches intractable. In particular, with a large number of states it is not possible to learn estimates of the expected future reward for each state, unless we can exploit commonalities between different states. The feature-based nature of ISU state representations expresses exactly these commonalities between states through the features that the states share. There are a number

```

STATE 13
DIALOGUE LEVEL
Turn: user
Speaker: user
ConvDomain: [about_task]
SpeechAct: [provide_info]
AsrInput: <date_time> october three first late morning</date_time>
TransInput: <date_time> october thirty first late morning</date_time>

TASK LEVEL
Task: [depart_time]
FilledSlotValue: [late morning]
FilledSlot: [depart_time]
CommonGround: [dest_city]

LOW LEVEL
WordErrorRate: 20.00

HISTORY LEVEL
SpeechActsHist: [], opening_closing, [], opening_closing, instruction, request_info,
[provide_info], implicit_confirm, request_info, [provide_info], implicit_confirm,
request_info, [provide_info]
TasksHist: [], meta_greeting_goodbye, [], meta_greeting_goodbye, meta_instruct, orig_city,
[orig_city], orig_city, dest_city, [dest_city], dest_city, depart_arrive_date, [depart_time]
FilledSlotsHist: [], [], [orig_city], [dest_city], [depart_time]
FilledSlotsValuesHist: [], [], [hartford connecticut], [orlando florida], [late morning]
GroundedSlotsHist: [], [], [orig_city], [dest_city]

```

Figure 1: Example fields from an Information State annotation. User information is in square brackets.

of techniques that could be used for RL with feature-based representations of states, but the simplest and most efficient is linear function approximation.

We use linear function approximation to map from a vector of real valued features  $f(s)$  for the state  $s$  to a vector of estimates  $Q(s, a)$  for each  $a$ . The trained parameters of the linear function are a vector of weights  $w_a$  for each action  $a$ . Given weights trained on a given dataset, an estimate  $Q_{data}(s, a)$  of the expected future reward given a state  $s$  and an action  $a$  is the inner product of the state vector  $f(s)$  and the weight vector  $w_a$ <sup>2</sup>.

$$Q_{data}(s, a) = f(s)^T w_a = \sum_i f_i(s) w_{ai}$$

The weights  $w_a$  are learnt from data, but the mapping  $f(s)$  from states to vectors must be specified beforehand. Each value  $f_i(s)$  in these vectors represents a possible commonality between states, so it is through the definition of  $f(s)$  that we control the notion of commonality which will be used by the linear function approximation. The definition of  $f(s)$  we are currently using is a straightforward mapping from feature-value pairs in the information state  $s$  to values in the vector  $f(s)$ .

The state vector mapping  $f(s)$  is computed using the first four levels of our annotation of the COMMUNICATOR data. We went through these annotations and identified the features which we consider relevant for dialogue management. These features were of three types. For features which take numbers

<sup>2</sup>We will use the notation  $x^T y$  to denote the inner product between vectors  $x$  and  $y$  (i.e. “ $x$  transpose times  $y$ ”).

as values, we used a simple function to map these numbers to a real number between 0 and 1, with the absence of any value being mapped to 0. For features which can have arbitrary text as their values, we used 1 to represent the presence of text and 0 to represent no value. The remaining features all have either a finite set of possible values, or a list of such values. Features with a list value are first converted to a list of pairs consisting of the feature and each value. For every possible feature-value pair, we define an element of the vector  $f(s)$  which is 1 if that feature-value pair is present in the state and 0 if it is not. These form the vast majority of our features. In total there are 291 features.

To train the weights of the linear approximation, we employed a standard RL learning method called SARSA( $\lambda$ ) [Sutton and Barto, 1998]. One advantage of using linear function approximation is that the learning method can be kept fairly simple, while still incorporating domain knowledge in the design of the mapping to feature vectors. One area of future research is to investigate more complicated mappings to feature vectors  $f(s)$ . This would bring us into the current research topic of kernel-based methods. Kernels are used to compensate for the over-simplicity of linear functions, and can be used to express more complicated notions of commonality between states [Shawe-Taylor and Cristianini, 2004].

### 3.3 Applying RL to a Fixed Dataset

We initially tried using the estimate of expected future reward  $Q_{data}(s, a)$  discussed in the previous section to define our dialogue policy. The dialogue policy simply selected the action  $a$  with the highest  $Q_{data}(s, a)$  given the state  $s$ . However, we found that this policy was very different from the policy ob-

served in the COMMUNICATOR data, almost never choosing the same action as was in the data. This simply means that the actions which have been learnt to have the best future rewards are not the ones that were typically chosen by the COMMUNICATOR systems in those states. Such actions would then lead to states unlike anything observed in the data, making the estimates for these states highly unreliable. In addition, the future reward depends on the policy the system uses in the future, so if the policy is different from that observed in the data, then the estimate  $Q_{data}(s, a)$  is not even relevant.

The solution to these problems which is typically used in RL research is to generate new data as learning progresses and the policy changes. The RL system can thus explore the space of possible policies and states, generating new data which is relevant to each explored policy and its states. Such policy exploration is often considered an integral part of RL. In future research, we intend to perform this exploration by running each policy with a user simulation (as in [Scheffler and Young, 2002]) trained on the COMMUNICATOR dataset, but first we need a solution to the problem of applying RL to a fixed set of data. Such policy exploration is only feasible with simulated dialogues generated through interaction with a simulated user, because generating real data with human users is very expensive. But a simulated user is not the same as a human user, so it is important to learn as much as possible from the fixed set of data we have from human users. In addition, the huge policy space makes even policy exploration with simulated users intractable, unless we can initialise the system with a good policy and constrain the exploration. This also requires learning as much as possible from the fixed set of data available before exploration.

There have been some proposals for learning a policy which is different from that used to generate the data (called off-policy learning), but these methods have been found not to work well with linear function approximation [Sutton and Barto, 1998]. They also do not solve the problem of straying from the region of state space which has been observed in the data.

### 3.4 A Hybrid Approach to RL

To address the above problems, we have investigated a novel hybrid approach which combines RL with supervised learning. The supervised learning is used to model the policy which the systems in the data actually use, which we model as a probabilistic policy  $S_{data}(s, a)$ .

$$S_{data}(s, a) \approx P(a|s)$$

In other words,  $S_{data}(s, a)$  is an estimate of the probability that a random system selected from those which generated the data would choose action  $a$  given that it is in state  $s$ . The function  $S_{data}(s, a)$  is computed with linear function approximation, just like  $Q_{data}(s, a)$ , except that a normalised exponential function is used so that the result is a probability distribution over actions  $a$ .

$$S_{data}(s, a) = \frac{\exp(f(s)^T w'_a)}{\sum_{a'} \exp(f(s)^T w'_{a'})}$$

As with the Q-function, the use of linear function approximation means that we have estimates for  $P(a|s)$  even for states  $s$

which have never occurred in the data, based on similar states which did occur.

The hybrid approach we have investigated is based on the assumption that we can't model the expected future reward for states in the unobserved portion of the state space. Thus we simply specify a fixed reward for these unobserved states. By setting this fixed reward to a low value, it amounts to a penalty for straying from the observed portion of the state space. The expected future reward is then the average between the fixed reward  $U$  for the cases where performing  $a$  in  $s$  leads to an unobserved state and the expected reward  $Q_{data}(s, a)$  for the cases where it leads to an observed state. Formally, this average is a mixture of the fixed reward  $U$  for unobserved states with the  $Q_{data}(s, a)$  estimate for observed states, where the mixture coefficient is the probability  $P_{observed}(s, a)$  that performing  $a$  in  $s$  will lead to an observed state.

$$\begin{aligned} E[\sum_{i>j} r(d, i)|s_j, a] \\ \approx Q_{data}(s, a)P_{observed}(s, a) + U(1 - P_{observed}(s, a)) \end{aligned}$$

Because this estimate of the expected future reward is only needed for choosing the next action given the current state  $s$ , we only need to estimate a function which discriminates between different actions in the same way as this estimate. To derive such a discriminant function, we first approximate  $P_{observed}(s, a)$  in terms of the probability distribution in the data  $P(s, a)$  and the size of the dataset  $N$ , under the assumption that the number of possible state-action pairs is much larger than the size of the dataset (so  $P(s, a)N \ll 1$ ).

$$\begin{aligned} P_{observed}(s, a) &= 1 - (1 - P(s, a))^N \\ &\approx P(s, a)N \approx S_{data}(s, a)P(s)N \end{aligned}$$

Given this approximation, the discriminant function needs to order two actions  $a_1, a_2$  in the same way as the above estimate of the expected future reward.

$$\begin{aligned} Q_{data}(s, a_1)S_{data}(s, a_1)P(s)N + U(1 - S_{data}(s, a_1)P(s)N) \\ \leq Q_{data}(s, a_2)S_{data}(s, a_2)P(s)N + U(1 - S_{data}(s, a_2)P(s)N) \end{aligned}$$

if and only if

$$S_{data}(s, a_1)(Q_{data}(s, a_1) - U) \leq S_{data}(s, a_2)(Q_{data}(s, a_2) - U)$$

We call this discriminant function  $Q_{hybrid}(s, a)$ .

$$Q_{hybrid}(s, a) = S_{data}(s, a)(Q_{data}(s, a) - U)$$

We use this  $Q_{hybrid}(s, a)$  function to choose the actions for our hybrid policy. By adjusting the value of the unobserved state penalty  $U$ , we can adjust the extent to which this model follows the supervised policy defined by  $S_{data}(s, a)$  or the reinforcement learning policy defined by  $Q_{data}(s, a)$ . In particular, if  $U$  is very low, then maximising  $Q_{hybrid}(s, a)$  is equivalent to maximising  $S_{data}(s, a)$ .

## 4 Experimental Results

We evaluate the trained dialogue management policies by running them against trained user simulations. Both the policies and the user simulations were trained using the annotated COMMUNICATOR data for the ATT, BBN, CMU, and SRI

systems. We compare our results against the performance of these same four systems, using an evaluation metric discussed below. The information states for the simulated dialogues were computed with the same rules used to compute the information states for the annotated data.

#### 4.1 The Testing Setup

For these experiments, we restrict our attention to users who only want single-leg flight bookings. This means there are only 4 essential slots to be filled: origin city, destination city, departure date, and departure time. To achieve this restriction, we first selected all those COMMUNICATOR dialogues which did not contain trip continuations.<sup>3</sup> This subset contained 79 BBN dialogues, 132 CMU dialogues, 258 ATT dialogues, and 174 SRI dialogues. This subset was used for evaluating the systems and for training the user model. The system model was trained on the full set of dialogues, since it should not know the user's goals in advance.

The user model was trained in the same way as the supervised component of the hybrid system discussed above, using linear function approximation and a normalised exponential output function. The states which precede user actions are input as vectors of features very similar to those used for the system but tailored to the needs of a user model. The output of the model is a probability distribution over actions, which consist of Speech Act, Task pairs. The user simulation selects an action randomly according to this distribution. We also trained a user model based on n-grams of user and system actions, which produced similar results in our testing.

When we first tested the hybrid policy, we found that it never closed the dialogue. We think that this is due to the system action (annotated in DATE) "meta\_greeting\_goodbye", which is used both as the first action and as the last action of a dialogue. The hybrid policy expects this action to be chosen before it will close the dialogue, but the system never chooses this action at the end of a dialogue because it is so strongly associated with the beginning of the dialogue. This is an example of the limitations of linear function approximation, which we plan to address by splitting this action into two actions, one for "greeting" and one for "goodbye". In the meantime, we have augmented the hybrid policy with a rule which closes the dialogue after the system chooses the action "offer", to offer the user a flight. We have also added rules which close the dialogue after 100 states (i.e. total of user and system actions), and which release the turn if the system has done 10 actions in a row without releasing the turn.

#### 4.2 The Evaluation Metric

To evaluate the success of a dialogue, we take the final state of the dialogue and use it to compute a scoring function. We want the scoring function to be similar to the reward we compute from the quality measures provided with the COMMUNICATOR data, but because we do not have these quality measures for the simulated dialogues, we cannot use the exact

<sup>3</sup>There are only 54 dialogues which contain continuations. Excluding these dialogues does not harm the evaluation of the COMMUNICATOR systems, since their average score is actually lower than that for the non-continuation dialogues (20.9).

same reward function. When we compare the hybrid policy against the COMMUNICATOR systems, we apply the same scoring function to both types of dialogues so that we have a comparable evaluation metric for both.

Because currently we are only considering users who only want single-leg flight bookings, the scoring function only looks at the four slots relevant to these bookings: origin city, destination city, departure date, and departure time. We give 25 points for each slot which is filled, plus another 25 points for each slot which is also confirmed (i.e. grounded). We also deduct 1 point for each action performed by the system, to penalise longer dialogues. Thus the maximum possible score is 198 (i.e. 200 minus 2 system actions: ask for all the user information in one turn, and then offer a flight).

The motivation behind this evaluation metric is that confirmed slots are more likely to be correct than slots which are just filled. If we view the score as proportional to the probability that a slot is filled correctly, then this scoring assumes that confirmed slots are twice as likely to be correct. When combining the scores for different slots, we do not try to model the all-or-nothing nature of the COMMUNICATOR task-completion quality measures, but instead sum the scores for the individual slots. This sum makes our scoring system value partial completions more highly, but inspection of the distributions of scores indicates that this difference does not favour either the hybrid policy or the original COMMUNICATOR systems.

Although this evaluation metric could reflect the relative quality of individual dialogues more accurately, we believe it provides a good measure of the relative quality of different systems. First, the exact same metric is applied to every system. Additional information which we have for some systems, but not all, such as the COMMUNICATOR user questionnaires, is not used. Second, the systems are being run against approximately equivalent users. The user simulation is trained on exactly the same user actions which are used to evaluate the COMMUNICATOR systems, so the user simulations mimic exactly these users. In particular, the simulation is able to mimic the effects of a speech recognition errors, since it is just as likely as the real users to disagree with a confirmation or provide a new value for a previously filled slot. The nature of the simulation model may make it systematically different from real users in some way, but we know of no argument for why this would bias our results in favour of one system or another.

#### 4.3 Comparisons Between Systems

We have run experiments to answer two questions. First, in our hybrid policy, what is the best balance between the supervised policy and the reinforcement learning policy? Second, how well does the hybrid policy perform compared to the COMMUNICATOR systems that it was trained on?

We trained models of both  $Q_{data}(s, a)$  and  $S_{data}(s, a)$ , and then used them in hybrid policies with various values for the unobserved state penalty  $U$ . For both functions, we trained them for 100 iterations through the training data, at which point there was little change in the training error. During testing, each hybrid policy was run for 1000 dialogues against the linear function approximation user model. The final state

$U$	total score	filled slots	grounded slots	length penalty
-1000	114.2	89.1	47.4	-22.2
0	101.3	69.5	51.6	-19.7
40	100.6	69.9	51.9	-21.1
80	96.0	67.0	49.4	-20.4
vs Ngram:				
80	105.7	68.8	57.2	-20.3

Table 2: The average scores for different values of the unobservable state reward  $U$ , and the three components of these scores.

for each one of these dialogues was then fed through the scoring function and averaged across dialogues. The results are shown in table 2. The values for  $U$  were chosen based on the average number of decisions per dialogue which were different from that which the purely supervised policy would pick, which were 0 ( $U = -1000$ ), 1 ( $U = 0$ ), 2 ( $U = 40$ ), and 5 ( $U = 80$ ), respectively.

Table 2 also shows some results for running the hybrid system against a user simulation based on n-grams of actions (“vs Ngram”). This user model seems to be easier to interact with than the linear user model. In particular, the resulting dialogues are better in terms of grounding.

To evaluate how well the hybrid policy performs compared to the COMMUNICATOR systems, we extracted the final states from all the non-continuation dialogues and fed them through the scoring function. The average scores are shown in tables 3 and 4, along with the best performing hybrid policy and the scores averaged over all systems’ dialogues.

Table 3 shows the results computed from the complete dialogues. These results show a clear advantage for the hybrid policy over the COMMUNICATOR systems. The hybrid policy fills more slots and does it in fewer steps. Because the number of steps is doubtless affected by the hybrid policy’s built-in strategy of stopping the dialogue after the first flight offer, we also evaluated the performance of the COMMUNICATOR systems if we also stopped these dialogues after the first flight offer, shown in table 4. The COMMUNICATOR systems do better when stopped at the first flight offer, but the ordering of the systems is the same. They do better on length, but worse on grounding, and about the same on filled slots.

#### 4.4 Discussion of Results

These results provide clear answers to both our questions, for this corpus and this approach to dialogue management. First, the more the system relies on the policy determined with supervised learning, the better it does. Second, the learnt policies perform better than any of the COMMUNICATOR systems.

The results in table 2 show a clear trend whereby the more the system sticks with the supervised policy, the better it does. In other words, the best the hybrid policy can do is simply mimic the typical behaviour it observes in the systems in the data. This is a surprising result, in that we were expecting reinforcement learning to provide some improvement over the supervised policy, provided the hybrid policy wasn’t too dif-

System	total score	filled slots	grounded slots	length penalty
hybrid RL/SL	114.2	89.1	47.4	-22.2
BBN	67.5	77.2	59.2	-68.9
CMU	49.7	60.2	47.4	-57.9
ATT	39.5	55.6	33.1	-49.3
SRI	21.0	52.4	0.0	-31.4
combined COMM	40.0	58.4	30.3	-48.6

Table 3: The average scores for the different systems, and the three components of these scores.

System	total score	filled slots	grounded slots	length penalty
hybrid RL/SL	114.2	89.1	47.4	-22.2
BBN	83.2	74.1	24.1	-15.0
CMU	63.9	55.1	26.9	-18.1
ATT	55.3	55.8	24.4	-25.0
SRI	27.8	52.2	0.0	-24.4
combined COMM	53.0	56.9	18.3	-22.2

Table 4: The average scores after the first flight offer for the different systems, and the three components of these scores.

ferent from the supervised policy. This may reflect the fact that RL is harder than supervised learning, or that the amount of data we are using isn’t enough to train an RL system effectively for such a complicated dialogue management task. We are currently producing better quality annotations, for the full set of COMMUNICATOR systems, which should improve the estimates for the RL component of the hybrid system. Another approach would be to only use the RL component for specific types of decisions (thereby simplifying the dialogue management task for RL). For example, the results in table 2 suggest that reinforcement learning improves grounding, but perhaps ends the dialogue before all the slots are filled.

It is worth noting that, even though the best system is purely supervised, the hybrid policies which do use some RL also do quite well. This can be seen by comparing their scores to the results for the COMMUNICATOR systems in tables 3 and 4. This performance is achieved despite the extreme difficulty of our state and policy spaces, as indicated by the very bad performance we observed when we initially tried a purely RL system. Therefore, we conclude that this hybrid approach would allow the fruitful use of RL in many situations where RL would otherwise be inappropriate due to the complexity of the task or the amount of data available. We anticipate that even for our very complex task, RL can be made beneficial by increasing the amount of data through simulated dialogues, which we intend to do in future work.

Tables 3 and 4 show a clear advantage of the learnt policy “hybrid RL/SL” over all the COMMUNICATOR systems. This result is perhaps surprising, since the learnt policy shown is the purely supervised version, which simply mimics the typical behaviour of these same systems. One likely explanation is that the hybrid policy represents a kind of multi-version system, where decisions are made based on what the majority

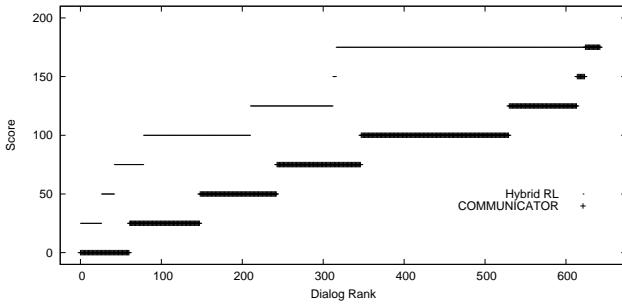


Figure 2: Comparing slot scores for the TALK learnt policy (“Hybrid RL”) versus the COMMUNICATOR systems (disregarding penalties for dialogue length)

of systems would do. Multi-version systems are well known to perform better than any one system alone, because the mistakes tend to be different across the different systems.

For a more detailed comparison of the systems, figure 2 plots the scores (ignoring length) for the average COMMUNICATOR dialogues versus the learnt policy’s simulated dialogues, as a function of their rank. Because the COMMUNICATOR systems do better when stopped after the first flight offer, we use these results for these plots. In figure 2, the length of each bar reflects how many dialogues achieved the associated score for the number of filled and/or grounded slots. None of the dialogues actually get the maximum score of 200 (all slots filled and grounded), but very many of the learnt policy dialogues score 175, compared to very few of the COMMUNICATOR dialogues reaching that score.

## 5 Conclusion

The learnt policy scores 37% higher than the best COMMUNICATOR system which we examined (114.2 versus 83.2). These are extremely promising results for a learnt policy with 70 actions, over  $10^{87}$  possible states, and very few hand-coded policy decisions. They indicate that linear function approximation is a viable approach to the very large state spaces produced by the ISU framework. They also show that this method for combining supervised learning with reinforcement learning is effective at learning policies in extremely large policy spaces, even with the limited amount of data in any fixed dataset. In the case of only using supervised learning, the linear function approximation is able to merge the policies of the systems in the COMMUNICATOR data and perform better than any one of these systems alone. Currently, adding reinforcement learning to this model degrades performance slightly, but still does better than any of the COMMUNICATOR systems. Further improvement should be possible by tailoring the representation of states and actions based on our experience so far.

The next step is to better exploit the advantages of reinforcement learning. One promising approach is to apply RL while running the learnt policy against simulated users, thereby allowing RL to explore parts of the policy and state spaces which are not included in the COMMUNICATOR data. The hybrid policy we have learnt on the COMMUNICATOR

data is a good starting point for this exploration. Also, the supervised component within the hybrid system can be used to constrain the range of policies which need to be explored when training the RL component. All of these advances will improve techniques for bootstrapping and automatic optimisation of dialogue management policies from limited initial datasets.

## Acknowledgements

This work is funded by the EC under the FP6 project “TALK: Talk and Look, Tools for Ambient Linguistic Knowledge”. We thank Johanna Moore for proposing the use of the COMMUNICATOR dataset for this work.

## References

- [Bos *et al.*, 2003] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo, 2003.
- [Cheyer and Martin, 2001] Adam Cheyer and David Martin. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148, 2001.
- [Georgila *et al.*, 2005] Kallirroi Georgila, Oliver Lemon, and James Henderson. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL), DIALOR’05*, 2005.
- [Larsson and Traum, 2000] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340, 2000.
- [Levin and Pieraccini, 1997] E. Levin and R. Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of Eurospeech*, pages 1883–1886, Rhodes, Greece, 1997.
- [Levin *et al.*, 2000] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.
- [Scheffler and Young, 2002] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. HLT*, 2002.
- [Shawe-Taylor and Cristianini, 2004] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- [Singh *et al.*, 2002] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, 2002.
- [Sutton and Barto, 1998] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.

[Walker and Passonneau, 2001] M. Walker and R. Passonneau. DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems. In *Walker, M., Passonneau R., DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems. In Proceedings of Human Language Technology Conference, San Diego, March, 2001.*, 2001.

[Walker *et al.*, 2001a] M Walker, J Aberdeen, J Boland, E Bratt, J Garofolo, L Hirschman, A Le, S Lee, S Narayanan, K Papineni, B Pellom, B Polifroni, A Potamianos, P Prabhu, A Rudnicky, G Sanders, S Seneff, D Stallard, and S Whittaker. Darpa communicator dialog travel planning systems: The june 2000 data collection. In *Eurospeech 2001*, Aalborg, Scandinavia, 2001.

[Walker *et al.*, 2001b] Marilyn A. Walker, Rebecca J. Passonneau, and Julie E. Boland. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *Meeting of the Association for Computational Linguistics*, pages 515–522, 2001.

[Walker *et al.*, 2002] M. Walker, A. Rudnicky, R. Prasad, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard. Darpa communicator: Cross-system results for the 2001 evaluation. In *ICSLP 2002*, 2002.

# Factored Partially Observable Markov Decision Processes for Dialogue Management

**Jason D. Williams**

Engineering Department  
Cambridge University  
Cambridge, UK  
jdw30@cam.ac.uk

**Pascal Poupart**

School of Computer Science  
University of Waterloo  
Ontario, Canada  
ppoupart@cs.uwaterloo.ca

**Steve Young**

Engineering Department  
Cambridge University  
Cambridge, UK  
s.jy@eng.cam.ac.uk

## Abstract

This work shows how a dialogue model can be represented as a factored Partially Observable Markov Decision Process (POMDP). The factored representation has several benefits, such as enabling more nuanced reward functions to be specified. Although our dialogue model is significantly larger than past work using POMDPs, experiments on a small testbed problem demonstrate that recent optimisation techniques scale well and produce policies which outperform a traditional fully-observable Markov Decision Process. This work then shows how a dialogue manager produced with a POMDP optimisation technique may be directly compared to a hand-crafted dialogue manager. Experiments on the testbed problem show that automatically generated dialogue managers outperform several hand-crafted dialogue managers, and that automatically generated dialogue managers for the testbed problem successfully adapt to changes in speech recognition accuracy.

## 1 Introduction

Creating (and improving) a dialogue manager by hand is typically an expensive and time-consuming undertaking. Instead of expressing which actions a machine should take in each dialogue situation, ideally a dialogue designer would simply express the desired *outcomes* of a dialogue. This specification would then be combined with a user model using a planning and optimisation algorithm to produce a dialogue manager. Markov Decision Processes (MDPs) provide a principled framework for this type of approach. The application of MDPs to the dialogue management problem is first explored by Levin and Pieraccini [1997]. Levin *et al.* [2000] provide a formal treatment of how a MDP may be applied to dialogue management, and Singh *et al.* [2002] show application to real systems.

MDPs assume the current state of the environment (i.e., the conversation) is known exactly and do not naturally or precisely model “noisy” evidence from the speech recogniser. This limitation has prompted several dialogue management researchers to explore POMDPs, which naturally express uncertainty in the current state. Roy *et al.* [2000] compare an MDP and a POMDP version of the same spoken dialogue system, and find that the POMDP version

gains more reward per unit time than the MDP version. Further, the authors show a trend that as speech recognition accuracy degrades, the margin by which the POMDP outperforms the MDP increases. Zhang *et al.* [2001] extend this work in several ways. First, the authors add “hidden” system states to account for various types of dialogue trouble, such as different sources of speech recognition errors. Second, the authors use Bayesian Networks to combine observations from a variety of sources (e.g., parse score, acoustic confidence score, etc.)

Looking outside the (PO)MDP framework, Paek and Horvitz [2003] suggest using a dynamic influence diagram to model user and dialogue state, and selecting actions based on “Maximum Expected [immediate] Utility.” This proposal can be viewed as a POMDP that greedily selects actions – i.e., which selects actions based only on immediate reward.<sup>1</sup> By choosing appropriate utilities, the authors show how local grounding actions can be automatically selected in a principled manner. In this work, we are interested in POMDPs as they enable planning over any horizon.

In previous work which has applied POMDPs to dialogue management, three important issues are not addressed. First, it is unclear in these models how to estimate the system dynamics in practice. For example, Zhang *et al.* [2001] indicate that the system dynamics are “handcrafted, depending a lot on the experience of the developer.” Second, neither model includes a notion of “dialogue state,” and as a result, the reward functions in these models cannot capture the notion of “appropriateness” of an action – for example, the relative appropriateness of confirming vs. querying a slot value.<sup>2</sup> Finally, although handcrafted dialogue managers are often used as a baseline comparison in dialogue system literature, the authors do not attempt a comparison with a handcrafted dialogue manager.

This paper makes two contributions. First, we propose a factored architecture for describing a POMDP-based dialogue manager. Unlike past work applying POMDPs

<sup>1</sup> We can express this formally as a POMDP with discount  $\gamma = 0$ . See section 2 for background on POMDPs.

<sup>2</sup> Zhang *et al.* [2001] included unobservable states for possible causes of dialog trouble – for example, “channel errors.” By contrast, in this work, we’re interested in the conventional sense of “dialogue state” as viewed by the user – for example, which items have been confirmed.

(and MDPs) to dialogue management, our factored representation adds a component for the state of the dialogue *from the perspective of the user*, enabling dialogue designers to add reward measures for the “appropriateness” of system actions. The factored representation also creates separate distributions for the user model and the speech recognition model, which facilitates estimating or adapting the system dynamics from dialogue data. Although the scope of our model results in a much larger model than past POMDP work on the dialogue management problem, we show (using a simple testbed problem) that the recently developed *Perseus* algorithm [Spaan and Vlassis, 2004] scales sufficiently to optimize our model and finds a policy which outperforms an MDP baseline.

Second, we show how to make direct comparisons between a hand-crafted and an automatically generated policy. We demonstrate this technique by introducing three hand-crafted dialogue managers for the testbed problem, and find that a dialogue manager created with an automated technique outperforms all of them.

The paper is organised as follows. Section 2 briefly reviews background on POMDPs. Section 3 presents the factored architecture. Section 4 shows an example testbed system using this architecture. Section 5 compares the testbed system to an MDP baseline, and assesses robustness in the face of changing speech recognition accuracy. Section 6 shows how a handcrafted policy can be compared to an automatically-generated policy, and makes this comparison for the testbed problem. Section 7 concludes.

## 2 Overview of POMDPs

Formally, a POMDP is defined as a tuple  $\{S, A_m, T, R, O, Z\}$ , where  $S$  is a set of states,  $A_m$  is a set of actions that an agent may take,<sup>3</sup>  $T$  defines a transition probability  $p(s'|s, a_m)$ ,  $R$  defines the expected (immediate, real-valued) reward  $r(s, a_m)$ ,  $O$  is a set of observations, and  $Z$  defines an observation probability,  $p(o'|s', a_m)$ .

The POMDP operates as follows. At each time-step, the machine is in some unobserved state  $s$ . The machine selects an action  $a_m$ , receives a reward  $r$ , and transitions to (unobserved) state  $s'$ , where  $s'$  depends only on  $s$  and  $a_m$ . The machine receives an observation  $o'$  which is dependant on  $s'$  and  $a_m$ . Although the observation gives the system some *evidence* about the current state  $s$ ,  $s$  is not known exactly, so we maintain a distribution over states called a “belief state,”  $b$ . We write  $b_t$  to indicate the distribution over all states at time  $t$ , and  $b_t(s)$  to indicate the probability of being in a particular state  $s$  at time  $t$ . The immediate reward is computed as the expected reward over belief states:

$$\rho(b_t, a_{mt}) = \sum_{s \in S} b_t(s)r(s, a_{mt}). \quad (1)$$

The goal of the machine is to maximise the cumulative, infinite-horizon, discounted reward called the *return*:

---

<sup>3</sup> In the literature, the system action set is often written as an un-subscripted  $A$ . In this work, we will model both machine and user actions, and have chosen to write the machine action set as  $A_m$  for clarity.

$$\sum_{t=0}^{\infty} \gamma^t \rho(b_t, a_{mt}) = \sum_{t=0}^{\infty} \gamma^t \sum_{s \in S} b_t(s)r(s, a_{mt}). \quad (2)$$

where  $\gamma$  is a geometric discount factor,  $0 \leq \gamma \leq 1$ . At each time step, the next belief state  $b'(s')$  can be computed exactly as shown in Eq. 12 below.

Because belief space is real-valued, an optimal infinite-horizon policy may consist of an arbitrary partitioning of  $S$ -dimensional space. In fact, the size of the policy space grows exponentially with the size of the observation set and doubly exponentially with the distance (in time-steps) from the horizon [Kaelbling *et al.*, 1998]. Nevertheless, real-world problems often possess small policies of high quality.

In this work, we make use of a recent *approximate* method called *Perseus*. *Perseus* [Spaan and Vlassis, 2004] is capable of rapidly finding good yet compact policies (when they exist). *Perseus* heuristically selects a small set of representative belief *points*, and then iteratively applies value updates to just those points, instead of all of belief space, thereby achieving a significant speed-up. *Perseus* has been tested on a range of problems, and found to outperform a variety of other methods, including grid-based methods [Spaan and Vlassis, 2004].

## 3 Factored architecture

Our proposal is to formulate the Dialogue Manager of a Spoken Dialogue System as a factored POMDP as follows.

First, the POMDP state variable  $s \in S$  is separated into three components: (1) the user’s goal,  $s_u \in S_u$ ; (2) the user’s action,  $a_u \in A_u$ ; and (3) the state of the dialogue,  $s_d \in S_d$ . The POMDP state  $s$  is given by the tuple  $\{s_u, a_u, s_d\}$ . We note that, from the machine’s perspective, all of these components are unobservable.

The user’s goal,  $s_u$ , gives the current goal or intention of the user. Examples of a complete user goal include a travel itinerary, a request for information about a calendar, or a product the user would like to purchase.

The user’s action,  $a_u$ , gives the user’s most recent user’s *actual* action. Examples of user actions include specifying a place the user would like to travel to, responding to a yes/no question, or a “null” response indicating the user took no action.

The state of the dialogue  $s_d$  indicates any relevant dialogue state information from the perspective of the user. For example,  $s_d$  might indicate that a particular slot has not yet been stated, has been stated but not grounded, or has been grounded.  $s_d$  enables a policy to make decisions about the appropriateness of behaviours in a dialogue – for example, if there are ungrounded items, a dialogue designer might wish to penalise asking an open question (vs. grounding an item).

Note that we do not include a state component for *confidence* associated with a particular user goal. The concept of confidence is naturally captured by the distribution of probability mass assigned to a particular user goal in the belief state.<sup>4</sup>

---

<sup>4</sup> Future work will explore how a speech recognition confidence score can be incorporated in a principled way.

The POMDP action  $a_m \in A_m$  is the action the machine takes in the dialogue. For example, machine actions might include greeting the user, asking the user where they want to go “to”, or confirming that the user wants to leave “from” a specific place. The POMDP observation  $o$  is drawn from the same set as  $a_u$ , i.e.,  $o \in A_u$ . Note that at each time step the POMDP receives a single observation, but maintains a distribution over all possible user actions.

To factor the model, we decompose the POMDP transition function as follows:

$$\begin{aligned} p(s' | s, a_m) &= p(s'_u, s'_d, a'_u | s_u, s_d, a_u, a_m) & (3) \\ &= p(s'_u | s_u, s_d, a_u, a_m) \cdot \\ &\quad p(a'_u | s'_u, s_u, s_d, a_u, a_m) \cdot \\ &\quad p(s'_d | a'_u, s'_u, s_u, s_d, a_u, a_m). & (4) \end{aligned}$$

We then assume conditional independence as follows. The first term – which we call the *user goal model* – indicates how the user’s goal changes (or does not change) at each time step. We assume the user’s goal at a time step depends only on the previous goal and the machine’s action:

$$p(s'_u | s_u, s_d, a_u, a_m) = p(s'_u | s_u, a_m). \quad (5)$$

The second term – which we call the *user action model* – indicates what actions the user is likely to take at each time step. We assume the user’s action depends on their (current) goal and the preceding machine action:

$$p(a'_u | s'_u, s_u, s_d, a_u, a_m) = p(a'_u | s'_u, a_m). \quad (6)$$

The third term – which we call the *dialogue model* – indicates how the user and machine’s actions affect the state of the conversation. We assume the current state of the dialogue depends on the previous state of the dialogue, the user’s action, and the machine’s action:

$$p(s'_d | a'_u, s'_u, s_u, s_d, a_u, a_m) = p(s'_d | a'_u, s_d, a_m). \quad (7)$$

In sum, our transition function is given by:

$$\begin{aligned} p(s' | s, a_m) &= p(s'_u | s_u, a_m) \cdot \\ &\quad p(a'_u | s'_u, a_m) \cdot \\ &\quad p(s'_d | a'_u, s_d, a_m). & (8) \end{aligned}$$

This factored representation reduces the number of parameters required for the transition function, and allows groups of parameters to be estimated separately. For example, we could estimate the *user action model* from a corpus by counting user dialogue acts given a machine dialogue act and a user goal, or use a “generic” distribution and adapt it to a particular problem once data becomes available.<sup>5</sup> We could then separately specify the dialogue model using a handcrafted function such as “Information State” update rules as in for example [Larsson and Traum, 2000].

The observation function is given by:

$$p(o' | s', a_m) = p(o' | s'_u, s'_d, a'_u, a_m). \quad (9)$$

---

<sup>5</sup> To appropriately cover all of the conditions, the corpus would need to include variability in the strategy employed by the machine – for example, using a Wizard-of-Oz framework with a simulated ASR channel [Stuttle *et al.*, 2004].

The observation function accounts for the corruption introduced by the speech recognition engine, so we assume the observation depends only on the action taken by the user:<sup>6</sup>

$$p(o' | s'_u, s'_d, a'_u, a_m) = p(o' | a'_u) = p(o | a_u). \quad (10)$$

The observation function can be estimated from a corpus or derived analytically using a phonetic confusion matrix, language model, etc. The observation can be discrete (i.e., a recognition hypothesis), or a mixture of discrete and continuous (i.e., a recognition hypothesis and a confidence score). Figure 1 shows an influence diagram of our proposal.

The reward function is not specified explicitly in this proposal since it depends on the design objectives of the target system. We note that the reward measure could contain incentives for dialogue speed (by using a per-turn penalty), appropriateness (through rewards conditioned on dialogue state), and successful task completion (through rewards conditioned on the user’s goal). Weights between these incentives could be estimated through formalisms like PARADISE [Walker *et al.*, 2000], and then adapted to the needs of a particular domain – for example, accuracy in performing a financial transaction is arguably more important than accuracy when obtaining weather information.

Finally, we update the belief state at each time step by:

$$\begin{aligned} b'(s') &= p(s' | o', a_m, b) \\ &= \frac{p(o' | s', a_m, b) p(s' | a_m, b)}{p(o' | a_m, b)} \\ &= \frac{p(o' | s', a_m) \sum_{s \in S} p(s' | a_m, b, s) p(s | a_m, b)}{p(o' | a_m, b)} \\ &= \frac{p(o' | s', a_m) \sum_{s \in S} p(s' | a_m, s) b(s)}{p(o' | a_m, b)} \quad (11) \end{aligned}$$

The numerator consists of the observation function, transition matrix, and current belief state. The denominator is independent of  $s'$ , and can be regarded as a normalisation factor; hence:

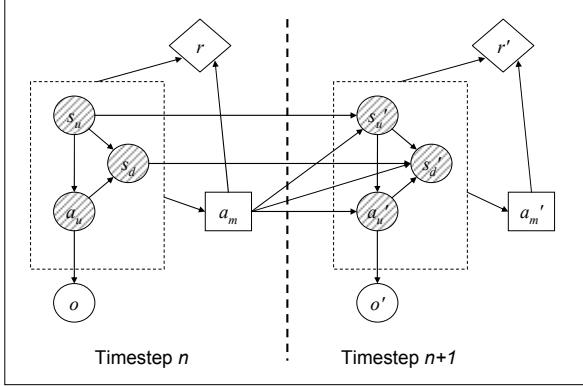
$$b'(s') = k \cdot p(o' | s', a_m) \sum_{s \in S} p(s' | a_m, s) b(s). \quad (12)$$

Substituting equation (8) and (10) into (12) and simplifying, we can write:

---

<sup>6</sup> This implicitly assumes that the same recognition grammar is always used. The model could be readily extended to enable a system “action” which activates a particular grammar.

$$\begin{aligned}
b'(s'_u, s'_d, a'_u) = & k \cdot p(o' | a'_u) p(a'_u | s'_u, a_m) \cdot \\
& \sum_{s_u \in S_u} p(s'_u | s_u, a_m) \cdot \\
& \sum_{s_d \in S_d} p(s'_d | a'_u, s_d, a_m) \cdot \\
& \sum_{a_u \in A_u} b(s_u, s_d, a_u). \tag{13}
\end{aligned}$$



**Figure 1: Influence diagram for the factored model.** The dotted box indicates the composite state  $s$  is comprised of three components,  $s_u$ ,  $s_d$ , and  $a_u$ . Shading indicates a component is unobservable. Arcs into circular chance nodes and diamond-shaped utility nodes show influence, whereas arcs into square decision nodes are informational (see Jensen [2001], p140).

#### 4 Testbed spoken dialogue system

To test the ideas in our proposal, we created a simulated dialogue management problem in the travel domain in which the user is trying to buy a ticket to travel from one city to another city. The machine asks the user a series of questions, and then “submits” the ticket purchase request, ending the dialogue. The machine may also choose to “fail”. In the testbed problem, there are three cities,  $\{a, b, c\}$ .

The machine has 16 actions available, including *greet*, *ask-from/ask-to*, *conf-to-x/conf-from-x*, *submit-x-y*, and *fail*, where  $x, y \in \{a, b, c\}, x \neq y$ . As above, the state space is given by the tuple  $\{s_u, a_u, s_d\}$ . The user’s goal  $s_u \in S_u$  specifies the user’s desired itinerary. There are a total of 6 user goals, given by  $s_u \in (x, y)$ ,  $x, y \in \{a, b, c\}, x \neq y$ . The dialogue state  $s_d$  contains three components. Two of these indicate (from the user’s perspective) whether the *from* place and *to* place have not been specified (*n*), are unconfirmed (*u*), or are confirmed (*c*). A third component *z* specifies whether the current turn is the first turn (*1*) or not (*0*). There are a total of 18 dialogue states, given by:

$$s_d \in (x_d, y_d, z); \quad x_d, y_d \in \{n, u, c\}, z \in \{1, 0\} \tag{14}$$

The user’s action  $a_u \in A_u$  and the observation  $o \in A_u$  are drawn from the set *x*, *from-x*, *to-x*, *from-x-to-y*, *yes*, *no*, and *null*, where  $x, y \in \{a, b, c\}, x \neq y$ .

These state components yield a total of 1944 states, to which we add one additional, absorbing end state. When the machine takes the *fail* action or a *submit-x-y* action, control transitions to this end state, and the dialogue ends.

The initial (prior) probability of the user’s goal is distributed uniformly over the 6 user goals. In the testbed

problem the user has a fixed goal for the duration of the dialogue, and we define the *user goal model* accordingly.

We define the *user action model* to include a variable set of responses – for example: the user may respond to *ask-to/ask-from* with *x*, *to-x/from-x*, or *from-x-to-y*; the user may respond to *greet* with *to-y*, *from-x*, or *from-x-to-y*; the user may respond to *confirm-to-x/confirm-from-x* with *yes/no*, *x*, or *to/from-x*; and at any point the user might not respond (i.e., respond with *null*). The probabilities in the user action model were chosen such that the user usually provides cooperative but varied responses, and sometimes doesn’t respond at all. The probabilities were handcrafted, selected based on the authors’ experience performing usability testing with slot-filling dialogue systems.<sup>7</sup> In future work, we intend to estimate a user model based on dialogue data.

We define the *dialogue model* to deterministically implement the notions of dialogue state above – i.e., a field which has not been referenced by the user takes the value *n*; a field which has been referenced by the user exactly once takes the value *u*; and a field which has been referenced by the user more than once takes the value *c*.

We define the observation function to encode the probability of making a speech recognition error to be  $p_{err}$ , and define the observation function as:

$$p(o | a_u) = \begin{cases} 1 - p_{err} & \text{if } o = a_u \\ \frac{p_{err}}{|A_u| - 1} & \text{if } o \neq a_u \end{cases} \tag{15}$$

Below we will vary  $p_{err}$  to explore the effects of speech recognition errors.

The reward measure includes components for both task completion and dialogue “appropriateness”, including: a reward of -3 for confirming a field before it has been referenced by the user; a reward of -5 for taking the *fail* action; a reward of +10 or -10 for taking the *submit-x-y* action when the user’s goal is  $(x, y)$  or not, respectively; and a reward of -1 otherwise. The reward measure reflects the intuition that behaving inappropriately or even abandoning a hopeless conversation early are both less severe than getting the user’s goal wrong. The per-turn penalty of -1 expresses the intuition that, all else being equal, short dialogues are better than long dialogues.

The reward measure also assigned -100 for taking the *greet* action when not in the first turn of the dialogue. This portion of the reward function effectively expresses a design decision: the *greet* action may only be taken in the first turn. A discount of  $\gamma = 0.95$  was used for all experiments.

The *Perseus* algorithm requires two parameters: number of belief points, and number of iterations. Through experimentation, we found that 500 belief points and 30 iterations attained asymptotic performance for all values of  $p_{err}$ .

<sup>7</sup> Because of space limitations, the detail of this distribution isn’t shown here.

## 5 Testbed evaluation

### 5.1 Comparison with an MDP Baseline

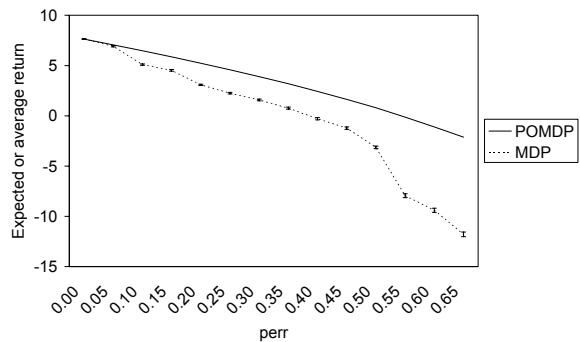
To test whether an automated solution to the POMDP is both feasible and worthwhile, we created an MDP-based dialogue manager baseline, patterned on systems in the literature (e.g., [Pietquin, 2004]). The MDP is trained and evaluated through interaction with a model of the environment, which is formed of the POMDP transition, observation, and reward functions. This model of the environment takes an action from the MDP as input, and emits an observation and a reward to the MDP as output.

The MDP state contains components for each field which reflect whether, *from the standpoint of the machine*, (a) a value has not been observed, (b) a value has been observed but not confirmed, or (c) a value has been confirmed. Two additional states – *dialogue-start* and *dialogue-end* – which were also in the POMDP state space, are included in the MDP state space for a total of 11 MDP states.

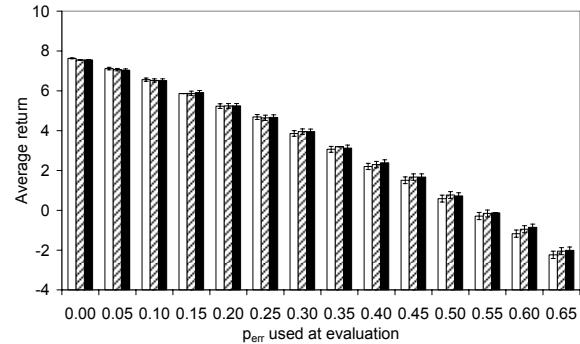
An MDP state estimator maps from POMDP observation to MDP state, and from MDP action to POMDP action. For example, given the current MDP state, the MDP policy selects an MDP action, and the MDP state estimator then maps the MDP action back to a POMDP action, which updates the environment model. The MDP state estimator tracks the most recent value observed for a slot, enabling it to map from an MDP action like *confirm-from* to a POMDP action like *confirm-from-a* or an MDP action like *submit* to *submit-from-a-to-b*. This behaviour of the MDP state estimator is identical to that used in the MDP/spoken dialogue system literature (e.g., [Pietquin, 2004] and [Levin et al., 2000]).

Because the MDP learns through experience with a simulated environment, we selected an on-line learning technique, Watkins Q-learning, to train the MDP baseline. A variety of learning parameters were explored, and the best-performing parameter set was selected: initial  $Q$  values set to 0, exploration parameter  $\epsilon = 0.2$ , and the learning rate  $\alpha$  set to  $1/k$  (where  $k$  is the number of visits to the  $Q(s,a)$  being updated.). To evaluate the resulting MDP policy, 10,000 dialogs were simulated using the learned policy.

Figure 2 shows expected return for the POMDP solution, and the average return for the MDP solutions vs.  $p_{err}$  ranging from 0.00 to 0.65. The (negligible) error bars show the 95% confidence interval for return assuming a normal distribution. Note that return decreases consistently as  $p_{err}$  increases for all solution methods, but the POMDP solution attains the largest return of the solutions at all values of  $p_{err}$ . Further, the performance gain of the POMDP solution over the other solutions increases as  $p_{err}$  increases. From this result we conclude that the POMDP solution copes with higher speech recognition error rates better than the MDP approach, consistent with [Roy et al., 2000].



**Figure 2: Expected or average return of POMDP policies and MDP baseline. Error bars show 95% confidence interval.**



**Figure 3: Performance of POMDP policies vs.  $p_{err}$ . White bars show a policy trained using  $p_{err}=0.15$ , checked bars  $p_{err}=0.35$ , and shaded bars  $p_{err}=0.55$ . Error bars show 95% confidence interval.**

### 5.2 Robustness to changes in error rate

In practice, the error rate of a spoken dialogue system varies from user to user. Thus we were interested to see how a POMDP solution adapts to a value of  $p_{err}$  for which it was not designed. Figure 3 shows average return for three dialogue managers when executed using a different value for  $p_{err}$ . Error bars show 95% confidence interval for true average return sampled over 10,000 dialogues. From this we see that the POMDP solutions are not “brittle” – i.e., they do not fail catastrophically as  $p_{err}$  deviates from that used in training.

## 6 Comparison with a handcrafted policy

### 6.1 Method to evaluate a handcrafted policy

Intuitively, a policy specifies what action to take in a given situation. In the previous section, we relied on the representation of a POMDP policy produced by value iteration – i.e., a value function, represented as a set of  $N$  vectors each of dimensionality  $|S|$ . We write  $v_n(s)$  to indicate the  $s$ th component of the  $n$ th vector.

Each vector represents the value, at all points in the belief space, of executing some “policy tree” which starts with an action associated with that vector. We write  $\hat{\pi}(n) \in A$  to indicate the action associated with the  $n$ th vector. If we assume that the policy trees have an infinite

horizon, then we can express the optimal policy at all timesteps as:

$$\pi(b) = \hat{\pi} \left( \arg \max_n \sum_{s=1}^{|S|} v_n(s) b(s) \right) \quad (16)$$

Thus the value-function method provides both a partitioning of belief space into regions corresponding to optimal actions as well as the expected return of taking that action. A second way of representing a POMDP policy is as a “policy graph” – a finite state controller consisting of  $N$  nodes and some number of directed arcs. Each controller node is assigned a POMDP action, and we will again write  $\hat{\pi}(n)$  to indicate the action associated with the  $n$ th node. Each arc is labelled with a POMDP observation, such that all controller nodes have exactly one outward arc for each observation.  $l(n, o)$  denotes the successor node for node  $n$  and observation  $o$ .

A policy graph is a general and common way of representing handcrafted dialogue management policies. More complex handcrafted policies – for example, those created with rules – can usually be compiled into a (possibly very large) policy graph.

A policy graph does not make the expected return associated with each controller node explicit. However, as pointed out by Hansen [1998], we can find the expected return associated with each controller node by solving this system of linear equations in  $v$ :

$$v_n(s) = r(s, \hat{\pi}(n)) + \gamma \sum_{s' \in S} \sum_{o \in O} p(s' | s, \hat{\pi}(n)) p(o | s', \hat{\pi}(n)) v_{l(n,o)}(s') \quad (17)$$

Solving this set of linear equations yields a set of vectors – one vector for each controller node. To find the expected value of starting the controller in node  $n$  and belief state  $b$  we compute:

$$\sum_{s=1}^{|S|} v_n(s) b(s) \quad (18)$$

## 6.2 Example handcrafted policies and results

Three handcrafted policies were created, called  $HC1$ ,  $HC2$ , and  $HC3$ . All of the handcrafted policies first take the action *greet*.  $HC1$  takes the *ask-from* and *ask-to* actions to fill the *from* and *to* fields, performing no confirmation. If the user does not respond, it re-trys the same action. If it receives an observation which is inconsistent or nonsensical, it re-trys the same action. Once it fills both fields, it takes the corresponding *submit-x-y* action. A logical diagram showing  $HC1$  is shown in Figure 4.<sup>8</sup>

$HC2$  is identical to  $HC1$  except that if the machine receives an observation which is inconsistent or nonsensical, it immediately takes the *fail* action. Once it fills both fields, it takes the corresponding *submit-x-y* action.

$HC3$  employs a similar strategy to  $HC1$  but extends  $HC1$  by confirming each field as it is collected. If the user

<sup>8</sup> A logical diagram is shown for clarity: the actual controller uses the real values  $a$ ,  $b$ , and  $c$ , instead of the variables  $X$  and  $Y$ , resulting in a controller with 15 states.

responds with “no” to a confirmation, it re-asks the field. If the user provides inconsistent information, it treats the new information as “correct” and confirms the new information. If the user does not respond, or if the machine receives any nonsensical input, it re-tries the same action. Once it has successfully filled and confirmed both fields, it takes the corresponding *submit-x-y* action.

Figure 5 shows the expected return for the handcrafted policies and the optimised POMDP solution. The POMDP solution outperforms all of the handcrafted policies for all values of  $p_{err}$ .

We inspected the POMDP solution in order to characterise how it differs from the handcrafted solutions. Conceptually, the POMDP policy differs from the handcrafted policies in that it tracks conflicting evidence rather than discarding it. For example, whereas the POMDP policy can interpret the “best 2 of 3” observations for a given slot, the handcrafted policies can maintain only 1 hypothesis for each slot.

As an illustration, consider an environment with no uncertainty – i.e., no speech recognition errors. In this environment, there is no benefit to maintaining multiple hypotheses for a user goal, and thus we would expect a POMDP to perform identically to a policy which does not track multiple hypotheses for a user goal. Figure 5 demonstrates this point: where  $p_{err} = 0$ ,  $HC1$  and  $HC2$  perform identically to the POMDP policy.<sup>9</sup>

It is interesting to note that  $HC3$ , which confirms all inputs, performs least well for all values of  $p_{err}$ . For the reward function we have provided in the testbed system, requiring 2 consistent recognition results (the response to *ask* and the response to *confirm*) gives rise to longer dialogs which outweigh the benefit of the increase in certainty

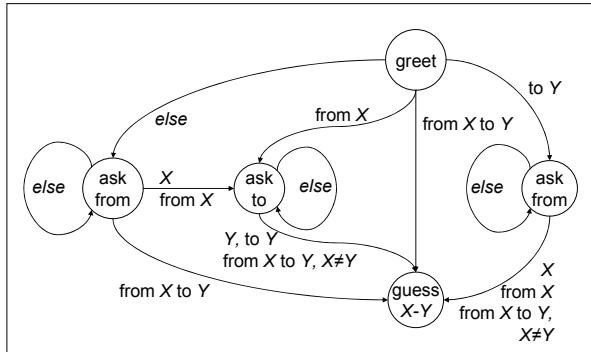
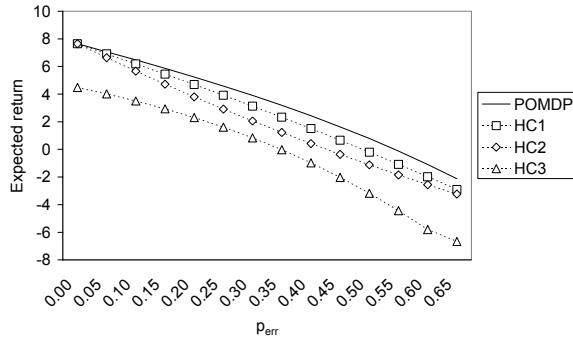


Figure 4:  $HC1$  handcrafted controller

<sup>9</sup>  $HC3$  performs worse because it confirms each element, lengthening the dialogue and thus reducing return.



**Figure 5: Expected return vs.  $p_{err}$  for POMDP policy and 3 handcrafted policies.**

## 7 Conclusion

We have proposed a factored architecture for describing POMDPs applied to spoken dialogue management. The factored representation is useful for two reasons – first, it facilitates estimating or specifying the system dynamics by reducing the number of parameters, and enabling different aspects of the system dynamics to be specified independently. Second, it enables incorporation of an explicit dialogue model from the user’s standpoint, which allows a dialogue designer to add rewards for “appropriate” dialogue behaviour. Further, we have shown how to convert a handcrafted policy represented as a finite-state controller into a value function, providing a principled way for handcrafted policies to be compared directly with policies produced with automated solutions. Our model is much larger than past POMDP dialogue managers; however, using our testbed problem, we have shown that a recent POMDP optimisation technique finds policies which outperform both an MDP baseline and three handcrafted controllers over all operating conditions. Further, the POMDP solution appears to adapt to changes in speech recognition error rate well.

A crucial theoretical issue is how to scale this model to handle larger problems since the state, action, and observation sets grow exponentially with the number of concepts in the problem. Although we have not used the factored representation to assist the optimisation process in this work, it may be possible to exploit the factoring to make the optimisation algorithms more efficient.

## References

- [Hansen, 1998] Eric A. Hansen. Solving POMDPs by searching in policy space. In *Uncertainty in Artificial Intelligence*, Madison, Wisconsin. 1998.
- [Jensen, 2001]. Finn V. Jensen. *Bayesian Networks and Decision Graphs*. New York: Springer Verlang, 2001.
- [Kaelbling et al., 1998] Leslie Pack Kaelbling, Michael L. Littman and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, Vol. 101, 1998.
- [Larsson and Traum, 2000] Staffan Larsson and David Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 5(3–4):323–340, 2000.

[Levin et al., 2000] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A Stochastic Model of Human-Machine Interaction for Learning Dialogue Strategies. *IEEE Transactions on Speech and Audio Processing*, Volume 8, No. 1, 11-23, 2000.

[Levin and Pieraccini, 1997] Esther Levin and Roberto Pieraccini. A Stochastic Model of Computer-Human Interaction For Learning Dialogue Strategies. *Eurospeech*, Rhodes, Greece, 1997.

[Paek and Horvitz, 2000] Tim Paek and Eric Horvitz. Conversation as Action Under Uncertainty. In *Proc. Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, June 2000.

[Pietquin, 2004] Olivier Pietquin. *A Framework for Unsupervised Learning of Dialogue Strategies*. Ph D thesis, Faculty of Engineering, Mons, Belgium, 2004.

[Roy et al., 2000] Nicholas Roy, Joelle Pineau and Sebastian Thrun. Spoken Dialogue Management Using Probabilistic Reasoning. *Annual meeting of the the Association for Computational Linguistics (ACL-2000)*.

[Singh et al., 2002] Satinder Singh, Diane Litman, Michael Kearns and Marilyn Walker. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence*, Vol. 16, 105-133, 2002.

[Spaan and Vlassis, 2004] Matthijs T. J. Spaan and Nikos Vlassis. *Perseus*: randomized point-based value iteration for POMDPs. Technical Report IAS-UVA-04-02, Informatics Institute, University of Amsterdam, 2004.

[Stuttle et al., 2004] Matthew Stuttle, Jason D. Williams, and Steve Young. A Framework for Wizard-of-Oz Experiments with a Simulated ASR-Channel. *International Conferences on Spoken Language Processing (ICSLP-2004)*, Jeju, South Korea, 2004.

[Walker et al., 2000] Marilyn A. Walker, Candace Kamm, and Diane Litman. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, Vol. 6, No. 3, 2000.

[Zhang et al., 2001] Zhang Bo, Cai Qingsheng, Mao Jianfeng, and Guo Baining. Planning and Acting under Uncertainty: A New Model for Spoken Dialogue System. *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, San Francisco, USA, 2001.

# Reinforcement Learning Of Dialogue Strategies Using The User's Last Dialogue Act\*

Matthew Frampton and Oliver Lemon

HCRC, School of Informatics

Edinburgh University

s0231685@sms.ed.ac.uk,olemon@inf.ed.ac.uk

## Abstract

Previous attempts at using reinforcement learning to design dialogue strategies for spoken dialogue systems e.g. [Singh *et al.*, 2002; Pietquin and Renals, 2002] have included only ‘low-level’ information in state representations i.e. whether or not a slot (e.g. destination city) has been filled and the confidence score associated with any supplied value. We explore the benefits of adding limited ‘high-level’ contextual information, in this case the dialogue act of the last user utterance. A general concern with adding more information is that the size of the state space might increase to a degree where learning becomes intractable. We describe 3 experiments in this paper which involve learning dialogue strategies for a flight booking dialogue system, and which test the potential benefits of adding this ‘high-level’ information to the state representation. We also explore the use of different reward functions in learning. In the first experiment, adding the high level information does not result in a superior learned strategy. However, the second experiment demonstrates a first simple case in which including the ‘high-level’ information does result in a superior learned strategy, producing a 52% increase in average reward, and the third scales up the problem to 4 slots. Here a reward function that rewards only totally correct database queries (rather than also valuing partial correctness) is found to produce the best strategy.

## 1 Introduction

The idea of using *Markov Decision Processes (MDPs)* and *reinforcement learning (RL)* to automate the design of dialogue strategies for dialogue systems was first proposed by [Levin and Pieraccini, 1997]. In the experiments we describe here, a reinforcement learning program learns a dialogue strategy while a user simulation (incorporating a simple ASR error simulation) converses with a dialogue manager. The domain is flight-booking, and the aim for the dialogue manager is

\*The support of the TALK project (EC IST: 507802) is acknowledged. We thank the reviewers for their comments.

to obtain the user’s flight information i.e. *departure, destination, date of travel* and *preferred airline* before making a database query. Previous researchers, e.g. [Singh *et al.*, 2002; Pietquin and Renals, 2002; Scheffler and Young, 2001], have used state representations containing only ‘low-level’ information, i.e. whether or not a slot has been filled and the confidence score associated with any supplied value. In this paper, we therefore begin to investigate the benefits of adding ‘high-level’ contextual information from the “Information State Update” approach to dialogue management, e.g. [Larsson and Traum, 2000], in this case the dialogue act of the last user utterance. Experiments 1 and 2 compare the performance of strategies learned with and without the dialogue act of the last user utterance in the state representation. In experiment 3, we also investigate different reward functions and issues of scalability.

Section 2 introduces the representation of human-machine dialogues as a Markov Decision Process (MDP), and the use of reinforcement learning to then learn dialogue strategies. Section 3 contains a description of our basic experimental framework, and a detailed description of the reinforcement learning component and user simulation. Section 4 describes each of the experiments, and in section 5 we conclude and suggest future work.

## 2 Dialogue as a Markov Decision Process

A human-machine dialogue can be represented as a Markov Decision Process (MDP) in terms of states (dialogue contexts) and actions (dialogue moves or speech acts) if we assume the Markov property. This property holds of systems where the state  $s_{t+1}$  of the system at time  $t + 1$  depends exclusively on the state  $s_t$  at the immediately preceding time  $t$ , and on the action  $a_t$  taken by the system when in state  $s_t$ :

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P_T(s_{t+1}|s_t, a_t) \quad (1)$$

where  $P_T$  is the transition probability.

The MDP formalism can be described as follows:

1.  $s_t$  is the state of the system at time  $t$ . In the experiments described in this paper, the states are dialogue context vectors containing features that indicate whether a slot has been filled, and the confidence score associated with any supplied value. There may also be a feature for the dialogue act of the last user utterance.

2.  $a_t$  is the action performed by the system or user at time  $t$ . Actions are taken from the finite action set  $A = \{a_i\}$ , and in the experiments here include utterances and database queries.
3. The reinforcement signal,  $r_t$  is the immediate reward or cost of having performed action  $a_t$  when in state  $s_t$ .
4.  $\pi$  is the system's strategy, and is a function from states to actions. In other words it states which (dialogue) action to perform in any state (dialogue context). It takes a single state as its argument:  $\pi(s_j)$  defines which action  $a_i$  is to be performed when in state  $s_j$ .

Given this representation, we are faced with the problem of deciding which dialogue actions it is best to perform in different states. This we do by Reinforcement Learning (RL), which is a method of learning by *delayed* reward using *trial-and-error search*. These two properties appear to make RL techniques a good fit with the problem of automatically optimising dialogue strategies, because in task-oriented dialogue often the “reward” of the dialogue (e.g. successfully booking a flight) is not obtainable immediately, and the large space of possible dialogues for any task makes some degree of trial-and-error exploration necessary.

Given a state and a possible action in that state, the *Q-function* describes the total reward expected between taking that action and the end of the dialogue. Once this function is available, the optimal dialogue strategy is simply to choose the action which maximises the expected future reward for each state. The *Sarsa*( $\lambda$ ) algorithm is a well understood reinforcement learning algorithm that can be used to learn the Q-function - see [Sutton and Barto, 1998] for a complete description. The *Sarsa*( $\lambda$ ) algorithm updates the Q-value for a state-action pair based on the Q-values of the state-action pairs it leads to later in the same dialogue, as well as based on the final rewards received.

### 3 The Experimental Framework

Each experiment is executed using the DIPPER dialogue manager [Bos *et al.*, 2003], a Reinforcement Learning program (“RLearner” described below) producing system dialogue acts, and a Bayesian User Simulation, all communicating using the *Open Agent Architecture* (*OAA*) [Cheyer and Martin, 2001].

We now describe the details of the Reinforcement Learner and Bayesian User Simulation OAA agents.

#### 3.1 The RLearner: system actions and rewards

In a single exchange between the system (RLearner) and simulated user, the DIPPER dialogue manager will first call RLearner's main solvable - *doRLearning(State,PossibleActions,Reward,NextAction)* - with the first 3 variables instantiated. For a non-terminal system turn, the reward will be specified as “*systemTurnPenalty*”. This represents a negative value stored internally by RLearner, and so causes RLearner to learn strategies that do not prolong a dialogue longer than necessary. In the experiments described in this paper, the value of *systemTurnPenalty* was always  $-1$ . RLearner will update its value estimates for actions

(Q-values) internally and provide the dialogue manager with the next action to be taken in the  $4^{th}$  variable. Below is a list of the possible system actions in experiments 2 and 3, which include all of those in experiment 1 plus an additional ‘give help’ action. Note that  $n$  is the number of slots.

1. An open question e.g. ‘How may I help you?’
2. Ask the value for any of slots  $1\dots n$ .
3. Explicitly confirm any of slots  $1\dots n$ .
4. Ask for the  $n$ th slot whilst implicitly confirming slot value  $n - 1^1$  e.g. ‘So you want to fly from Edinburgh to where?’
5. Give help.
6. Database Query.

There are a couple of restrictions regarding which actions can be taken in which states. Action 1 (open question) is only possible at the start of the dialogue, and DIPPER can only confirm non-empty slots. The dialogue manager carries out the action in the  $4^{th}$  variable and then calls the user simulation's main solvable - *generateResponse(SystemPrompt, UserResponse, ConfidenceScores, AbstractResponse)* - with only the  $1^{st}$  variable instantiated. The user simulation (see section 3.3) returns a response and its abstract representation i.e. the dialogue act, and the dialogue manager updates the dialogue context accordingly.

This sequence of events repeats until the conclusion of the dialogue, and the system and user simulation may have any number of dialogues. A dialogue concludes when either the user simulation ‘hangs-up’ (with final reward 0), or the system makes a *database query*. In the case of a *database query*, the dialogue manager calculates a reward, which it passes to RLearner via the main solvable *doRLearning/4*. The reward functions explored in the experiments were the following:

1. Reward Function 1 (Partial Reward): +100 for each correct slot value.
2. Reward Function 2 (All-or-nothing Reward): IF all slot values are correct, +100, ELSE 0.
3. Reward Function 3 (Mixed Reward): IF all slot values are correct, +100, ELSE +10 for each correct slot value.

#### 3.2 RLearner parameters

When the RLearner agent is initialized, it is given a parameter string which includes the following:

1. Step Parameter:  $\alpha$
2. Discount Factor:  $\gamma$
3. Eligibility Trace Parameter:  $\lambda$
4. Eligibility Trace Type: *accumulating* or *replacing*.
5. Action Selection Type:  *$\epsilon$ -greedy* or *softmax*.
6. Action Selection Parameter:  $\epsilon$  if  *$\epsilon$ -greedy*; *temperature* if *softmax*.
7. Initial Q-values

---

<sup>1</sup>Where  $n = 1$  we implicitly confirm the final slot.

8. Print the *Q-values* and learned strategy to file every *X* dialogues.
9. Calculate the *average reward* over every *Y* dialogues.
10. Draw a graph of average reward against number of dialogues every *Z* dialogues.
11. System-Turn-Penalty: a negative reward

RLearner updates its Q-values using the *Sarsa*( $\lambda$ ) algorithm. As stated in section 2, see [Sutton and Barto, 1998] for a complete description of this method. The first parameter is the step-parameter  $\alpha$  which may be a value between 0 and 1, or specified as *decreasing*. If it is *decreasing*, as it is in our experiments, then for any given Q-value update  $\alpha$  is  $\frac{1}{k}$  where  $k$  is the number of times that the state-action pair for which the update is being performed has been visited. The 2<sup>nd</sup> and 3<sup>rd</sup> parameters,  $\gamma$  and  $\lambda$ , may take values between 0 and 1. In the experiments here, they are always 1 and 0.9 respectively. The 4<sup>th</sup> parameter specifies whether the eligibility traces are *replacing*, as they are here, or *accumulating*. Apart from updating Q-values, RLearner must also choose the next action for the dialogue manager to take and the 5<sup>th</sup> parameter specifies whether it does this by  *$\epsilon$ -greedy* or *softmax* action selection (in our experiments to date we have explored only the  *$\epsilon$ -greedy* case). The 6<sup>th</sup> parameter gives the value of  $\epsilon$  for  *$\epsilon$ -greedy* action selection when training, or the *temperature* if using softmax. In the experiments here,  $\epsilon$  is always 0.7, meaning that in 30% of cases a sub-optimal action is explored during training.

RLearner will output 1 file containing the Q-values and another containing the learned policy every *X* dialogues (parameter 8). A Q-values file can be loaded into RLearner and this makes it possible to continue learning from this point, or to test the learned strategy using RLearner's solvable *getLearnedAction(State,Action)*, which given the state, returns the action with the highest Q-value. RLearner is also able to draw graphs of average reward over different intervals of dialogues, and these graphs are useful for indicating when the learned strategy has stabilised. Finally, the 11<sup>th</sup> parameter is the system-turn penalty mentioned in section 3.1.

### 3.3 Bayesian User Simulation

Here a *user simulation*, rather than real users, interacts with the dialogue system during learning. This is because thousands of dialogues may be necessary to train even a simple system (here we train on up to 100,000 dialogues), and for a proper exploration of the state-action space, the system should sometimes take actions that are not optimal for the current situation, making it a sadistic and time-consuming procedure for any human training the system. [Eckert *et al.*, 1997] were the first to use a user simulation for this purpose, but it was not goal-directed and so could produce inconsistent utterances. The later simulations of [Pietquin and Renals, 2002] and [Scheffler and Young, 2001] were goal-directed and also incorporated an *ASR error simulation*. A user simulation interacts with the system via *intentions*, which are closely related to *speech acts* or *dialogue acts*. Intentions are preferred because they are easier to generate than word sequences and because they allow error modelling of all parts of the system, for example *ASR error modelling* and semantic errors.

The user and ASR simulations must be realistic if the learned strategy is to be directly applicable in a real system.

Our user simulation is capable of simulating goal-directed mixed-initiative dialogue. When the user simulation's main solvable *generateResponse* (*SystemPrompt*, *UserResponse*, *ConfidenceScores*, *AbstractResponse*) is called with the first variable instantiated, it first generates an abstract response via a *Bayesian Belief Network* (implemented using NeticaJ [NorsysCorp., 2002]). Figure 1 shows the structure of this Bayesian network. We chose this structure after analysing COMMUNICATOR data [Walker *et al.*, 2001] and identifying what seemed to be the most important factors in determining the next user response. These were the last system prompt, how many times each slot has been asked about, whether or not the system has attempted to confirm any slot values incorrectly, and whether the user has asked for help. An example of an incorrect confirmation caused by an ASR error is given in the dialogue segment below:

User Simulation: I want to fly from Edinburgh.  
System: So you want to fly from Eindhoven?

Whether the system has attempted to confirm any slot values incorrectly, and whether the user has asked for help is represented by the 'history' node. We supplied the probabilities based on an initial analysis of flight-booking dialogues, but in future work we expect to use probabilities learned from COMMUNICATOR data.

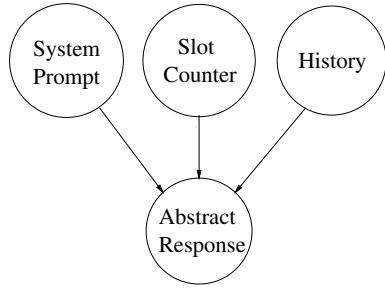


Figure 1: The Bayes Net used by the user simulation

The possible values for the *system prompt* node are the possible system actions given in section 3.1. The different possible values for the *slot counter* node are:

- The slot asked about in the most recent system turn has been asked about  $\leq 1$  time before.
- The slot asked about in the most recent system turn has been asked about  $\geq 2$  times before.

The different possible values for the *history* node are:

- Null: There are no outstanding incorrect system confirmations.
- The dialogue system has attempted to confirm slot *n* incorrectly.
- The user has asked for help.

Finally, the different possible values for the *abstract response* are:

1. Stay quiet.
2. Give the value for any of slots 1...n.
3. Give the values for all of the slots.
4. Say “no” and give the correct slot value in response to an incorrect system confirmation.
5. Say “no”.
6. Say “yes”.
7. Ask for help.
8. End the dialogue i.e. ‘hang-up’.

Abstract responses 1, 4 and 5 are responses given in utterance to an incorrect system confirmation. If the abstract response is 1, 4, or 5, then it will include the number of the slot which has been incorrectly confirmed. Having generated an abstract user response via the Bayesian net, the user simulation next translates this abstract response into a concrete response (*userResponse* in the *generateResponse* solvable) by referring to a random user goal that it selected at the beginning of the dialogue. This random goal consists of values for *departure city*, *destination city*, *date of travel* and *preferred airline*. Hence if the abstract response is to give the value for slot 1, and the goal departure city is ‘Edinburgh’, then the concrete user response (*userResponse*) will be of the form *departure(edinburgh)*. The user simulation may take the initiative by giving more than one slot value when it is asked for only one, and may decide to close the dialogue if asked the same question twice or more.

Next this concrete user response is subject to an ASR simulation. Our initial approach is that each word in the user simulation’s vocabulary belongs to a cluster which contains similar sounding words. At this stage we have built these clusters by hand, but in future we expect to use a full ASR error simulator – for example a stochastic acoustic confuser whose conditional probabilities are learned from the SACTI corpora [Williams and Young, 2004]. The larger a word’s cluster, the lower will be its ASR confidence score, and the greater the likelihood of an ASR error. An ASR error is simulated by substituting the word for another from the same cluster. The 2<sup>nd</sup> variable is then instantiated with the concrete response which now possibly contains a simulated ASR error, and the 3<sup>rd</sup> variable with the associated ASR confidence score(s). The ASR confidence score may be either *low* or *high*. If a filled slot has low ASR confidence and is unconfirmed, then there is only a 55% chance that the value is correct, while if its ASR confidence is high, then it is correct.

### 3.4 Discussion

The experiments here can be viewed as the first in a set of preliminary experiments which all use a user simulation based on a Bayesian net, the probabilities for which we supplied after an initial analysis of flight-booking dialogues. In future experiments we plan to use a user simulation whose probabilities are learned from COMMUNICATOR data [Walker *et al.*, 2001] with extended annotations for user dialogue acts [Georgila *et al.*, 2005]. The user and ASR simulations must be realistic if the learned strategy is to be directly applicable

in a real system. Nevertheless, we believe the preliminary experiments to be worthwhile, first of all because they are proving successful and so further motivate undertaking future experiments. They also provide insights into which contextual features it is important to include in the state representation and why. We are also gaining useful knowledge about how best to set the reinforcement learner’s parameters.

## 4 The Experiments

### 4.1 Experiment 1

Experiment 1 used a 2-slot dialogue system, where the 1<sup>st</sup> slot refers to the departure city, and the 2<sup>nd</sup> to the destination city. The user simulation was designed to ‘hang-up’ if the system confirms a slot value incorrectly and then on its next turn, having been alerted by the user simulation, it fails to re-ask or re-confirm this slot value. The possible system actions and user responses included those listed in sections 3.1 and 3.3 respectively, except for giving and asking for help. Reward function 2 (All-or-nothing Reward), described in section 3.1, was used, since in a series of preliminary experiments (not presented here), it had been found to produce faster learning and better strategies when using  $\epsilon$ -greedy action selection. The experiment involved learning a strategy without the dialogue act of the last user utterance in the state space (i.e. variables 1-6 shown below) (call this the “baseline” strategy), and then with the dialogue act of the last user utterance (variables 1-7) (the DA strategy), before comparing their relative performance.

Variables in the state-space:

1. Is the 1<sup>st</sup> slot filled?
2. Is the 2<sup>nd</sup> slot filled?
3. The ASR confidence score for the 1<sup>st</sup> slot value.
4. The ASR confidence score for the 2<sup>nd</sup> slot value.
5. Has the 1<sup>st</sup> slot value been confirmed?
6. Has the 2<sup>nd</sup> slot value been confirmed?
7. Dialogue act of the last user utterance.

Variables 1-6 are binary, and there were 10 different dialogue acts for the last user utterance, as listed here:

1. slot1
2. slot2
3. yes
4. no-repair(slot1)
5. no-repair(slot2)
6. no(slot1)
7. no(slot2)
8. quiet(slot1)
9. quiet(slot2)
10. hang-up

This makes a total of  $2^6 \times 10 = 640$  states. On first encountering this problem, including the dialogue act of the last user utterance in the state space might be expected to produce a

superior learned strategy, because if RLearner does not know that the last user utterance was a rejection of an attempted confirmation, then how can it learn to re-ask or reconfirm the slot value under question on its next turn?

However, the results showed no real difference in performance between the baseline and DA strategies. After 18000 dialogues, each of the learned strategies seemed to have stabilised, as shown by figure 2.

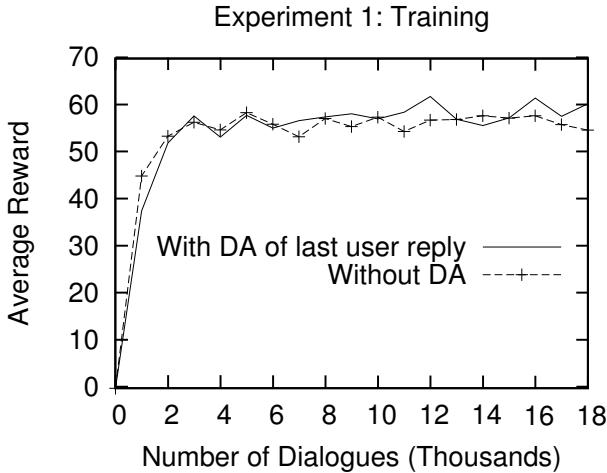


Figure 2: Learning in experiment 1: 2 slot system, reward function 2 (All-or-nothing Reward), no user “help” action.

Interacting with the user simulation during testing runs, the strategy learned with the DA-strategy then achieved an average reward-per-dialogue of 94.05 over 500 dialogues, and the baseline, 93.80.

### Analysis

Our intuition is that how a user rejects an attempted system confirmation ought to make certain system recovery actions more suitable than others, and then it would be beneficial to include the dialogue act of the last user utterance in the state space. However for the particular user simulation used in this experiment, ‘how?’ is not important, only ‘whether?’. As a result it is unnecessary to include the dialogue act of the last user utterance.

To see why let us consider examples of two different types of incorrect system confirmations. The first is where the relevant slot has yet to be confirmed. Consider the following state-action pair taken from the text file representing the learned strategy after 18000 dialogues.

```
State: [1, 1, low, low, no, no]
Action: explicitlyConfirmSlot1
```

If the explicit confirmation is incorrect, then the user simulation rejects it and the 5<sup>th</sup> variable does not turn to ‘no’ meaning the state stays the same. Thus the next system action is also to explicitly confirm slot 1. The system has not shifted its focus away from the problem 1<sup>st</sup> slot - something which would have caused the user simulation to hang-up. The second type of incorrect system confirmation is where the relevant slot has already been confirmed. In this case, an incor-

rect confirmation will change the state. Here is an example of two relevant state-action pairs:

```
State: [0, 1, low, low, no, yes]
Action: implicitlyConfirmSlot2andAskSlot1
```

```
State: [0, 1, low, low, no, no]
Action: implicitlyConfirmSlot2andAskSlot1
```

An incorrect implicit confirmation of the 2<sup>nd</sup> slot value will be rejected by the user simulation and we will move from the first to the second state. In this second state, RLearner has also learned to implicitly confirm the 2<sup>nd</sup> slot whilst asking for the 1<sup>st</sup>. Thus again, the system has not shifted its focus away from the problem slot.

Below is a list of further general characteristics of the learned strategies from this experiment. These characteristics are also true of the learned strategies from experiments 2 and 3.

1. An open question is asked at the start of the dialogue.
2. Slot values with low ASR confidence scores are confirmed (implicitly or explicitly) rather than re-asked.
3. When at least 1 but not all slots are filled, an empty slot is asked while a filled slot is implicitly confirmed.
4. Once all slots are filled, explicit is preferred to implicit confirmation for slot values with low ASR confidence.

RLearner learns to ask an open question at the start of a dialogue because the user simulation is more likely to reply with > 1 slot value, so enabling a shorter dialogue. To explain characteristic 2, we must consider slots that have been filled with low ASR confidence but are unconfirmed. The advantage of using confirmation over re-asking comes when the confirmation turns out to be correct, which is the majority of cases. Then the user simulation replies with ‘yes’ which is always recognised correctly, whereas if the system re-asks the slot value, then the user simulation replies with the value and this has a 45% chance of being misrecognised. Characteristic 3 appears because if an implicit confirmation is correct, then the system can confirm the slot and obtain a value for an empty slot in just one turn, and this allows shorter dialogues. However, asking for one slot value while implicitly confirming another is not desirable if all slots are filled because slot values provided by the user simulation may be incorrectly recognised and so cause longer dialogues and/or incorrect database queries. Hence, in these states the learned strategy chooses explicit confirmation (characteristic 4).

## 4.2 Experiment 2

The aim of experiment 2 was to investigate a first simple case in which adding the dialogue act of the last user utterance to the state space does allow RLearner to learn a superior strategy. As in experiment 1, we used a 2-slot dialogue system, reward function 2 (All-or-nothing Reward) from section 3.1, and the same variables for representing the state space. The key difference was that the user simulation was now able to ask for help, and the system was able to give help – hence the possible user responses and system actions included all those listed in sections 3.3 and 3.1 respectively. The user simulation had a 20% chance of asking for help if:

- The system has just asked an open question (this is only possible at the start of a dialogue).
- The system asks for slot 1 for the first time.
- The system asks for slot 2 for the first time.

The user simulation will not ask for help in any other situations, and it now closes the dialogue if it asks for help and then is not given help in the next system turn. There are now 11 different dialogue acts for the last user utterance - the same 10 from experiment 1 listed in section 4.1, and the eleventh, ‘ask-help’. With the dialogue act of the last user utterance in the state representation, there are now  $2^6 \times 11 = 704$  states. Again, two strategies were learned - the first including the dialogue act of the last user utterance in the state space, (the “DA-strategy”), and the second lacking this information (the baseline strategy). Learning lasted for 18000 dialogues after which both learned strategies were stable, as shown by figure 3.

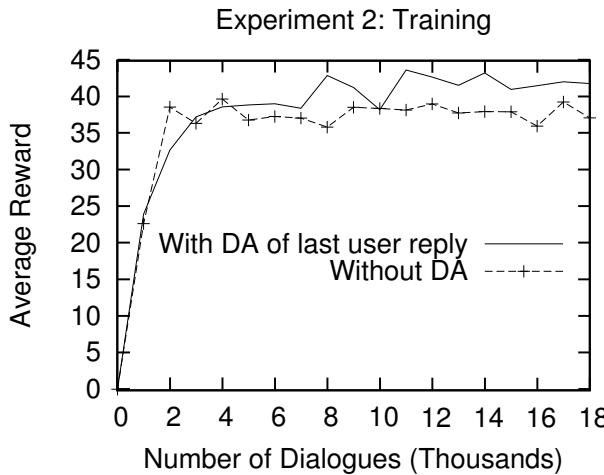


Figure 3: Learning in experiment 2: 2 slot system, reward function 2 (All-or-nothing Reward), with user ‘help’ action.

The DA-strategy gave help whenever the user asked for it, while the baseline strategy did not give help in any state. As a result, the DA-strategy achieved an average reward-per-dialogue of 90.55 over a 500 dialogue test run interacting with the user simulation, and the baseline strategy had an average reward of only 59.64 (i.e. the DA-strategy shows a 52% improvement). The rewards during testing are greater than in the training graph (figure 3) since in testing the system does not choose any sub-optimal actions (i.e.  $\epsilon = 1$ ). This experiment then successfully demonstrates a first simple case in which adding the dialogue act of the last user utterance allows RLearner to learn a superior strategy.

### 4.3 Experiment 3

To explore scaling-up, together with the impact of using different reward functions, in experiment 3 a 4-slot dialogue system was used, where slots 3 and 4 refer to the *date of travel*, and *preferred airline* respectively. Each state was represented in terms of the following variables:

1. Slot 1 confidence score.
2. Slot 2 confidence score.
3. Slot 3 confidence score.
4. Slot 4 confidence score.
5. The dialogue act of the last user utterance.

Variables 1-4 can take the values *empty*, *low* or *high*. If a value for the slot has been supplied, and if its ASR confidence score is low and it is unconfirmed, then the value for the variable will be *low*. If it is confirmed or if its ASR confidence score is *high*, then it will be *high*. There were 19 possible values for variable 5 (see the user simulation actions in section 3.3), and so a total of  $3^4 \times 19 = 1539$  states. Strategies 1, 2, and 3 were learned using each of the 3 reward functions listed in section 3.2. Learning continued for 100000 dialogues, at which stage the learned strategies were still improving, but the one using reward function 1 (Strategy 1: Partial Reward) at a slower rate than the 2<sup>nd</sup> and 3<sup>rd</sup> reward functions (All-or-nothing and Mixed Rewards, Strategies 2 and 3 respectively). This is shown in figures 4, 5, and 6.

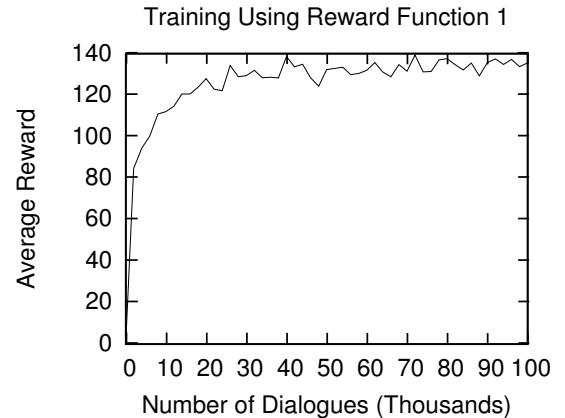


Figure 4: Learning in experiment 3: 4-slot system, reward function 1 (Partial reward), with user ‘help’ action,  $\epsilon$ -greedy action selection,  $\epsilon = 0.7$

Here we can see that the general dialogue strategy learning problem remained tractable for the 4 slot case, and that after 30000 and 100000 dialogues, with  $\epsilon$ -greedy action selection and  $\epsilon$  set to 0.7, the best strategy was learned using the 2<sup>nd</sup> reward function (All-or-nothing Reward), the 2<sup>nd</sup> best strategy using the 3<sup>rd</sup> reward function (Mixed Reward), and the worst one using the 1<sup>st</sup> reward function (Partial Reward). The experiment was repeated twice and this result was the same each time. The performance of the learned strategies from the first run are summarised in tables 1 and 2 (columns represent dialogue strategies trained with the 3 different reward functions, rows show testing of those strategies based on the 3 ways of computing reward). The numbers represent the average reward achieved per dialogue over 500 test dialogues.

When learning with reward function 1, RLearner has learned to query the database in many more states where < 4 slots have been filled with high confidence than when learning with reward functions 2 or 3. This is apparently because it

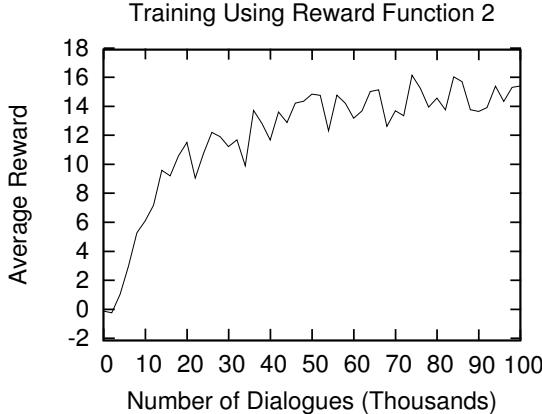


Figure 5: Learning in experiment 3: 4-slot system, reward function 2 (All-or-nothing Reward), with user ‘help’ action,  $\epsilon$ -greedy action selection,  $\epsilon = 0.7$

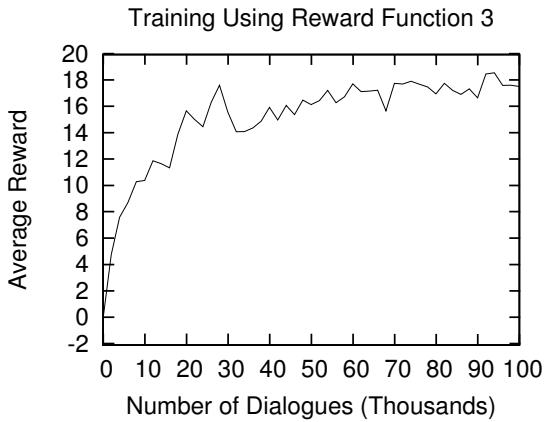


Figure 6: Learning in experiment 3: 4-slot system, reward function 3 (Mixed Reward), with user ‘help’ action,  $\epsilon$ -greedy action selection,  $\epsilon = 0.7$

still receives a substantial reward for doing so: +100, +200, or +300 rather than the maximum attainable +400. Note that learned strategy 2 outperformed learned strategies 1 and 3 on all performance metrics i.e. including reward functions 1 and 3.

This experiment fails to consider the fact that for a real spoken dialogue system there are situations where it is preferable to query the database with only a subset of the slots filled. For example, the ASR may simply be unable to recognise a particular slot value, perhaps due to the user’s accent. Then it is clearly better to query the database with the slot values that can be obtained rather than asking for the problem slot over and over again before the user becomes so frustrated that they hang-up. Our next experiments, which will look at the benefits of including a dialogue length feature in the state space will consider such situations. We will investigate whether we can learn superior strategies using *softmax* action selection rather than  $\epsilon$ -greedy.

		Strategy 1	Strategy 2	Strategy 3
Reward	1	247.89	373.90	334.06
Function	2	20.12	84.42	62.74
(Testing)	3	35.65	85.30	71.90

Table 1: Testing the 3 strategies after 100000 training dialogues using each of the 3 reward functions, average reward achieved per dialogue over 500 test dialogues.

		Strategy 1	Strategy 2	Strategy 3
Reward	1	186.76	320.04	301.91
Function	2	-4.54	78.69	40.32
(Testing)	3	14.58	75.16	56.79

Table 2: Testing the 3 strategies after 30000 training dialogues using each of the 3 reward functions, average reward achieved per dialogue over 500 test dialogues.

## 5 Conclusion and Future Work

We have explored reinforcement learning of full dialogue strategies using limited “high-level” context information (the user’s last dialogue act). Previous work (e.g. [Singh *et al.*, 2002; Pietquin and Renals, 2002; Scheffler and Young, 2001]) has focused on only “low-level” information such as the filled slots. Also, previous work [Singh *et al.*, 2002] concentrated only on specific strategy decisions (e.g. confirmation and initiative strategies), rather than the full problem of what system dialogue action to take next.

To summarise our results, in the first experiment, adding the dialogue act of the last user utterance to the state space does not produce a superior learned strategy. Since in this case *how* the user simulation rejected an incorrect confirmation was not important, only *whether* the information in the ‘low level’ state space was sufficient. The second experiment, where the system can respond to user “help” requests, demonstrates a first simple case in which including this ‘high-level’ information does result in a superior learned strategy (there is a 52% improvement in average reward). The third experiment then shows that we can scale up to a more commercially realistic 4-slot problem, and that when using  $\epsilon$ -greedy action selection, a reward function which rewards only totally correct database queries (All-or-nothing versus Partial Rewards) produced the best strategy.

In future work we intend to explore how strategies for different dialogue phenomena can be captured by adding incrementally more high-level context information, while scaling up further and maintaining tractability of learning (e.g. by using linear function approximation techniques [Henderson *et al.*, 2005; Lemon *et al.*, 2005]). For example we expect to learn confirmation, clarification, and backtracking strategies that are sensitive not only to ASR confidence scores and the user’s last dialogue move, but also to dialogue context features such as dialogue length, the system’s last dialogue move, parse ambiguity, user intention recognition, the available modalities, and so on. Our next experiment will explore more subtle confirmation strategies by adding dialogue length as a state feature.

## References

- [Bos *et al.*, 2003] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. Dipper: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, 2003.
- [Cheyer and Martin, 2001] Adam Cheyer and David Martin. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148, 2001.
- [Eckert *et al.*, 1997] Weiland Eckert, Esther Levin, and Roberto Pieraccini. User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.
- [Georgila *et al.*, 2005] Kallirroi Georgila, Oliver Lemon, and James Henderson. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: DIALOR) (to appear)*, 2005.
- [Henderson *et al.*, 2005] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems, (to appear)*, 2005.
- [Larsson and Traum, 2000] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340, 2000.
- [Lemon *et al.*, 2005] Oliver Lemon, Kallirroi Georgila, James Henderson, Malte Gabsdil, Ivan Meza-Ruiz, and Steve Young. D4.1: Integration of Learning and Adaptivity with the ISU approach. Technical report, TALK Project, 2005.
- [Levin and Pieraccini, 1997] Esther Levin and Roberto Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, Rhodes, Greece, 1997.
- [NorsysCorp., 2002] NorsysCorp. *Netica-J Manual*. Norsys Software Corp., 2315 Dunbar Street, Vancouver, Canada, June 2002.
- [Pietquin and Renals, 2002] Olivier Pietquin and Steve Renals. ASR system modeling for automatic evaluation and optimization of dialogue systems. In *International Conference on Acoustics Speech and Signal Processing, ICASSP*, Orlando, 2002.
- [Scheffler and Young, 2001] Konrad Scheffler and Steve Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *NAACL-2001 Workshop on Adaptation in Dialogue Systems*, Pittsburgh, USA, 2001.
- [Singh *et al.*, 2002] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, 2002.
- [Sutton and Barto, 1998] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.
- [Walker *et al.*, 2001] M Walker, J Aberdeen, J Boland, E Bratt, J Garofolo, L Hirschman, A Le, S Lee, S Narayanan, K Papineni, B Pellom, B Polifroni, A Potamianos, P Prabhu, A Rudnicky, G Sanders, S Seneff, D Stallard, and S Whittaker. Darpa communicator dialog travel planning systems: The june 2000 data collection. In *Eurospeech 2001*, Aalborg, Scandinavia, 2001.
- [Williams and Young, 2004] Jason Williams and Steve Young. The SACTI-1 corpus: Guide for research users. Technical report, University Of Cambridge, 2004.