Generation of collaborative spoken dialogue contributions in dynamic task environments

Oliver Lemon
Alexander Gruenstein
Randolph Gullett
Alexis Battle
Laura Hiatt
Stanley Peters
CSLI, Stanford University,
210 Panama Street,
Stanford, CA 94305
{lemon,alexgru,rgullett,ajbattle,lahiatt,peters}@csli.stanford.edu

Abstract

We explain generation methods in a dialogue system allowing humans and complex devices or applications to collaborate in real-time dialogue about ongoing activities in dynamic environments. The generation component must be able to handle contexts where there are multiple topics being co-ordinated by the conversation and where world and system-states can vary independently. We describe turn-management, truth-checking, "relevance"-checking, and the incremental message selection, aggregation, and generation methods employed in this context. We demonstrate that these techniques are viable in a demonstration dialogue system for multi-modal conversations with semi-autonomous mobile robots.

Introduction

Here we point out general problems in utterance generation for dialogue systems of a certain complexity (Allen *et al.* 2001; Lemon, Gruenstein, & Peters 2002). Our concern is a class of dialogues which co-ordinate activities in "dynamic task environments" – application domains where the system may complete, cancel, plan, or suspend multiple tasks, and where executing plans may fail. An unpredictable operating environment also means that objects of discussion may appear and disappear at any time. Contrast these sorts of contexts with the domains of typical "form-filling" dialogue systems supporting information-seeking dialogues – where the relevance, form, and timing, of generated utterances are largely predetermined.

We consider the core requirements on utterance generation for dialogue systems in these more complex situations to be:

- advance the dialogue whenever possible
- be relevant
- never make false statements[1]
- avoid mode confusion[2] on the part of the user
- manage the dialogue initiative (turn-taking)[3]
- avoid confusion about referring expressions where possible.

Generation for spoken dialogue systems, as opposed to text planning, is generally problematic in that dialogue contributions arise incrementally, often in response to another participant's utterances. For this reason, generation of large pieces of text is not appropriate, especially since the user should be able to interrupt the system. Other differences abound, for example that aggregation rules (see below) must be sensitive to incremental aspects of message generation during ongoing dialogue.

As well as the general problems of message selection and aggregation in dialogue systems, application domains with dynamic environments and changing device-states present specific problems in comparison with, say, travel-planning dialogue systems. For instance, such a device or service will in general need to communicate about,

- perceptions of a changing environment,
- progress towards user-specified goals,
- execution status of activities or tasks,
- its own internal state changes,
- the state of the dialogue itself.

For these reasons, the message selection and generation component of such a system needs to be of wider coverage and more flexible than template-based approaches, while remaining in real, or near-real, time (Stent 1999). As well as this, the system must be able to deal with a potentially large bandwidth stream of communications from the device[4], and so must be able to intelligently filter them for "relevance" so

[1] But, as a reviewer pointed out, systems should perhaps produce only approximately true statements when that would be helpful.

[2] The issue is that users might be misled with regard to the states of activities being carried out.

[3] For example, give the user time to respond to system utterances, such as questions, when appropriate.

[4] For example a mobile robot may make many observations per second, and it is not possible or desirable to report them all.

Figure 1: Demonstration system, september 2002

Utterance	**Dialogue Move**
Operator (O): *Where are you?*	Wh-question
Robot (R): *I am at the warehouse*	Wh-answer
O: *Always fly at high altitude*	Specify Global Constraint
R: *Okay*	Confirmation
O: *Pick up the medical supplies at the hospital*	Command
R: *Wait a minute*	Grab turn
R: *I am supposed to always fly high*	Report Global Constraint
R: *Shall I fly to the hospital at low altitude anyway?*	Negotiate constraint relaxation
O: *Yes. Do it anyway*	Yn-Answer
R: *Wait a minute*	Grab turn
R: *There is a fire at the tower*	Report observation
R: *I am flying to the hospital*	Report Current Activity
O: *Why?*	Why-question
R: *Because I am picking up the medical supplies there.*	Why-Answer
O: *Stop that*	Command
R: *Okay. I have cancelled flying to the hospital*	Report Activity State
O: *Fight the fire at the tower*	Command
R: *Okay. I am flying to the lake*	Report Activity State
O: *Why are you doing that?*	Why-question
R: *Because I am picking up the water there.*	Why-Answer
O: *What are your constraints?*	Wh-question
R: *I am supposed to always fly at high altitude.*	Wh-Answer

that the user is not overloaded with unimportant information or repetitious utterances.

The system must also have some knowledge about when it is appropriate to grab or release the turn (i.e. when to take and relinquish dialogue initiative), and of how to produce dialogue moves which have these effects (Traum & Allen 1994). In a previous version of our system, no turn-taking routine was implemented, and some very unhelpful dialogues resulted. For example the following (imagine almost no time delay between the system utterances):

```
User: Go to the building.
System: Which building do you mean?
System: I can see a blue car at the tower.
System: It is driving on Creek Lane.
System: Warning my fuel is low.
User: I mean the school.
```

Here, the system should be designed so that the user is be given a chance to answer its questions, unless there are more pressing concerns (e.g. emergency reports). It should also know that when the user is speaking they retain the turn unless they explicitly pass it to the system (e.g. by asking a question), and it should know how to grab the turn from the user ("excuse me ...", "just a minute ...") and when to do so. For this reason we implemented a system of *turn-management* described below.

Another concern is that in a complex dialogue system, there may be utterances that the system has queued to say (in our case, on the System Agenda) but which are no longer true when time has been found to say them. For instance, the system can queue a report about its own state or a world-state (e.g. "I am flying to the tower") which ceases to be true before time is found to say it. If the system goes ahead and utters the report, the user may be mislead into believing a false statement. For this reason – and to avoid mode confusion – we need to implement a system of *truth-checking* before queued system utterances are sent to the speech synthesizer.

In general, also, the system should appear as "natural" as possible from the user's point of view – using the same language as the user if possible ("echoing" and "alignment"), using anaphoric referring expressions where possible, and aggregating utterances where appropriate. A "natural" system should also exhibit "variability" in that it can convey the same content in a variety of ways. A further desirable feature, related to alignment, is that the system's generated utterances should be in the coverage of the dialogue system's speech recognizer, so that system-generated utterances effectively prime the user to speak in-grammar – see (Hockey *et al.* 2002; 2003).

Consequently we attempted to implement the following features in the message generation component of our dialogue system:

1. turn-management
2. truth-checking
3. relevance filter
4. recency filter (avoiding unnecessary repetition)
5. echoing user referring expressions
6. variability
7. aggregation
8. priming - using only "in coverage" utterances

9. real-time message generation.

The novel features of our message generation system derive from the rich dialogue context to which the generation module has access – in particular the "Activity Tree" (Lemon, Gruenstein, & Peters 2002) which represents the temporal and hierarchical structure of tasks which the back-end system is executing, has executed, and plans to execute, and their states (*current, planned, failed, suspended, cancelled, complete*). Many of the utterances which the system plans to generate are initiated due to monitoring the properties of the Activity Tree as it is constructed, through collaborative dialogue, by the user and the system. We describe this process further below.

Figure 1 shows the level of competence we have achieved in our current demonstration system, which is designed for performing collaborative tasks with a robotic helicopter (Lemon *et al.* 2001) (note: the robot decides to fly low to the hospital due to a general constraint that when picking up objects at a location it must fly at low altitude). Although the particular domain of this example involves a robotic helicopter, we have designed and constructed the system so that domain specific elements (e.g. tasks done by the robot or device) are separated from general conversational skills (e.g. turn-taking, dialogue move interpretation, generation).

Generation, LFs, and the System Agenda

Our basic generation method is to take as inputs to the process various communicative goals of the system, expressed as a list of logical forms (LFs) on the System Agenda, together with the dialogue state (e.g. salient objects, referring expressions, preceding utterance) and use them to construct new logical forms to be input to Gemini's Semantic Head-Driven Generation algorithm (Shieber *et al.* 1990), which produces strings for either Festival (Taylor, Black, & Caley 1998) or Nuance Vocalizer (Nuance 2003) speech synthesis.

Inputs to the generation module are "concept" logical forms describing the communicative goals of the system. These are structures consisting of *context tags* (e.g. activity identifier, turn tag) and a *content logical form* consisting of a Dialogue Move type (e.g. *report, wh-question*), a priority tag (e.g. *warn* or *inform*), and some additional content tags (e.g. for objects referred to). An example input logical form is,

```
report(inform, agent(AgentID), cancel-
activity(ActivityTag))
```

which for example could correspond to the report "I have cancelled flying to the tower" when `AgentID` refers to the system and `ActivityTag` refers to a "fly to the tower" task. Such logical forms are instantiated and placed on the System Agenda by way of monitoring properties of the Activity Tree.

Our basic generation method is shown in Figure 2 (the various tests and functions are explained in the remainder of the paper).

Items which the system will consider for generation are placed (either directly by the device, or indirectly by the Activity Tree) on the "System Agenda" (SA), which is the part of the dialogue Information State which stores communicative goals of the system. Communicative goals may also exist on the "Pending List" (PL) which is the part of the Information State which stores questions that the system has asked, but which the user has not answered, so that they may be re-raised by the system. Only questions previously asked by the system can exist on the Pending List. Questions can be removed from the Pending List by user dialogue moves such as "Forget (about) the X" where X is an NP mentioned in the pending question.

Note that other items, for example confirmations which have not yet been spoken, are not placed on the Pending List, but instead go on the System Agenda. The System Agenda is used for items that are queued up to be said as soon as possible, while the Pending List is a list of items that need to be re-added to the System Agenda unless they are dealt with.

Turn-management

We implemented a simple 3-valued turn system, where the turn marker can either be `System`, `User`, or `Neither`. Different dialogue moves then have different effects on the turn marker. For instance, questions always swap the turn, and answers always release it. Many dialogue moves have no effect on the turn, and the user is allowed the privileged position of being able to grab the turn from the system at any point simply by speaking (obviously this is not appropriate for many genres of dialogue, for example tutorial dialogues). As well as this, if either participant has the turn but fails to use it, they lose it after a time-out of a few seconds. Before the system can produce an utterance, it always checks whether or not it has the turn. If the user has the turn and an urgent message (one flagged as a "warning") needs to be generated, the system waits until the user is not speaking, produces a "grab turn" utterance, and then produces its utterance.

The demonstration system displays a turn marker on the GUI, allowing observers to watch the changing possession of the turn.

Truth-checking

As well as checking that it has the turn before generating an utterance, the system also checks that any statements it is about to make are (still) true. Of course, many dialogue contributions, such as questions, are neither true nor false, in which case this step does not apply. For assertive dialogue moves such as *report* and *answer* the system checks that the state they describe still obtains, with respect to its own representation of its ongoing activities and world-knowledge. This process uses the Activity Tree and theorem proving using the Java Theorem Prover – for details see (Lemon, Gruenstein, & Peters 2002).

Message selection – Relevance

Since a complex system may make many observations in a short time, and is potentially carrying out multiple activities

Figure 2: The Basic Generation algorithm

```
for U = Logical Form of preceding system utterance
    A = Logical Form at top of System Agenda
    B = next Logical Form on System Agenda

if A=U, remove A from System Agenda              \\ recency check

else if  (turn=system or turn=neither)           \\ check turn
     and A is Relevant
     and A is True
     then ReplaceNPs(A) to form A';               \\ echoing or anaphoric NPs
          Pre-aggregate(A',B) to form A'';
          Retro-aggregate(A'',U) to form A''';
          GeminiGenerate(A''', String);           \\ Gemini's SHDG
          TextToSpeech(String);                   \\ Festival or Nuance Vocalizer
          Remove A from System Agenda.
```

at once, a particular problem is how to determine appropriate generation of utterances about those activities and observations, in a way which does not overload the user with information, yet which establishes and maintains appropriate context in a natural way. This is why we implemented a *relevance filter* on utterance generation.

Due to multi-tasking considerations, at any time there may be a number of "Current Activities" which the user and system are collaboratively performing (e.g. fly to the tower, search for a red car). These activities are topics of conversation represented in the dialogue Information State, and the system's reports can be generated by them (in which case they are tagged with that activity label) or can also be *relevant* to an activity in virtue of being about an object which is in focus because it is involved in that activity.

Thus we define the class of "relevant" utterances to be those concerning planned or current activities, or objects involved in those activities, or those involving objects previously mentioned in the dialogue.

Some system reports are more urgent that others (e.g. "I am running out of fuel") and these carry the label *warning*, and always grab the turn. Warnings are always relevant, no matter what activities are current – they always pass the recency and relevance filters. If a warning is triggered while the system is formulating an utterance, the warning goes on top of the System Agenda. In fact, the probability of this happening is very low anyway, since messages only take in the order of a few milliseconds to formulate. In the more likely case where the warning is received for generation when the system is speaking, the system does not interrupt itself, but queues the warning at the top of the System Agenda.

Choosing NPs, Echoing

Echoing (for noun-phrases) is achieved by accessing the Salience List whenever generating referential terms, and using whatever noun-phrase (if any) the user has previously employed to refer to the object in question.

However, in generation of NPs, the system should use variation and anaphora only in as much as is required for naturalness which avoids misunderstanding. For instance, the system may refer to a red car that the user has spoken about either as "it" or "the car". If the system were always to use the phrase "a red car", users could be misled into believing that a new red car is being spoken about. Thus, when considering how to generate an NP, the system determines whether the object it refers to (denoted in the LF) has been spoken about before by the user, and whether it is in fact the most salient object in the dialogue at that point. If it is, the chosen referring expression will be anaphoric. If it is a new object, the system will introduce it with a descriptive phrase, and if it is a known object the system will modify the referring expression last used.

Incremental aggregation

Aggregation (Appelt 1985) combines and compresses utterances to make them more concise, avoid repetitious language structure, and make the system's speech more natural and understandable overall. Aggregation techniques on a prewritten body of text combine and compress sentences that have already been determined and ordered. In a complex dialogue system however, aggregation should produce similarly natural output, but must function incrementally because utterances are generated on the fly. In dialogue systems, when constructing an utterance we often have no information about the utterances that will follow it, and thus the best we can do is to compress it or "retro-aggregate" it with utterances that preceded it (see the example below).

Sometimes, however, the System Agenda contains enough unsaid utterances to perform reasonable "pre-aggregation".

Each dialogue move type (e.g. *report*, *wh-question*) has its own aggregation rules, stored in the class for that LF type. In each type, rules specify which other dialogue move types can aggregate with it, and exactly how aggregation works. The rules note identical portions of LFs and unify them, and then combine the non-identical portions appropriately.

For example, the LF that represents the phrase "I will fly to the tower and I will land at the parking lot", will be converted to one representing "I will fly to the tower and land at the parking lot" according to the reduction rules. Similarly, "I will fly to the tower and fly to the hospital" gets converted to "I will fly to the tower and the hospital".

In contrast, the "retro-aggregation" rules result in sequences of system utterances such as,

```
System:  I have cancelled flying to the
base
System:  and the tower
System:  and landing at the school.
```

Priming the user

The end result of our selection and aggregation module is a fully specified logical form which is then sent to the Semantic-Head-Driven Generation component of Gemini (Dowding *et al.* 1993; Shieber *et al.* 1990). The bi-directionality of Gemini (i.e. that we use the same grammar for both parsing and generation) automatically confers a useful "symmetry" or "alignment" property on the system – that it only utters sentences which it can also understand. This means that the user will not be misled by the system into employing out-of-vocabulary items, or out-of-grammar constructions. Thus, as desired, the system's utterances can prime the user to make in-grammar utterances.

Multi-modality

A final aspect to note is that the system is able to perform limited multi-modal generation using its map display (see `http://www-csli.stanford.edu/semlab/witas`) and a video output window. Whenever an object is mentioned in the spoken dialogue, its icon is highlighted on the map. In some cases, questions (e.g. "where is the tower?") are best answered with multimodal output, in which case the System Agenda is given a simple answer to say (e.g. "here you are") and the appropriate icon(s) are highlighted on the map display.

The exploration of multi-modal generation is an area for future work.

Summary

We argued that in the case of dialogues with complex systems in dynamic task environments, a generation mechanism has to be particularly sensitive and flexible. This is especially so in comparison with template-based generation in information-seeking dialogues, such as travel planning, where the relevance, form, and timing, of generated utterances is largely predetermined. Another challenge was that conversations may have several open topics at any one time, to which generated utterances may contribute. Thus the operational domain of such a dialogue system forces us to deal with the following issues:

- Turn-management
- Truth-checking
- Relevance checking - message selection
- Incremental aggregation
- Generation of "echoic" referring expressions

We discussed the representations and algorithms used to produce a generation component with the following features:

- echoic and variable message generation, filtered for relevance and recency,
- real-time generation,
- supports collaborative dialogue in dynamic task environments.

Evaluation

As part of the research described in (Hockey *et al.* 2002; 2003), we have performed only a limited evaluation of the system thus far. 20 volunteers, with no previous experience using the system, have each completed a session consisting of five tasks (an example task is "Search the area for a red truck. Follow it until it stops, then land back where you started"). Data has been collected regarding completion time, steps to completion, and speech recognition error rates. All dialogues have been recorded, and the Information States logged as HTML files.

In our experiments 45% of subjects abandoned the first task (including two that gave up on every task), while 20% of subjects abandoned their fifth task. Subjects finished the first task with an average time of 440.6 seconds, and by the fifth task their average time to completion was 267.4 seconds. See (Hockey *et al.* 2003) for the full details.

Future work

We aim to explore multi-modal generation, the use of prosodic mark-up, and adaptive generation with respect to user-models. Another area of research will be the use of reactive planning (Shapiro, Langley, & Shachter 2001) in utterance planning and generation within a multi-layer architecture for dialogue systems (Zinn, Moore, & Core 2002).

Acknowledgements

This research was (partially) funded under the Wallenberg laboratory for research on Information Technology and Autonomous Systems (WITAS) Project, Linköping University, by the Wallenberg Foundation, Sweden. We wish to thank Anne Bracy for her work on the Gemini grammar while at CSLI, Patrick Doherty of WITAS, and the referees for their comments.

References

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2001. Toward conversational human-computer interaction. *AI Magazine* 22(4):27–37.

Appelt, D. E. 1985. Planning english referring expressions. *Artificial Intelligence* 26(1):1 – 33.

Dowding, J.; Gawron, J. M.; Appelt, D.; Bear, J.; Cherny, L.; Moore, R.; and Moran, D. 1993. GEMINI: a natural language system for spoken-language understanding. In *Proc. 31st Annual Meeting of the ACL*.

Hockey, B.-A.; Aist, G.; Hieronymous, J.; Lemon, O.; and Dowding, J. 2002. Targeted help: Embedded training and methods for evaluation. In *Proceedings of Intelligent Tutoring Systems (ITS)*, 70 – 74.

Hockey, B. A.; Lemon, O.; Campana, E.; Hiatt, L.; Aist, G.; Hieronymus, J.; Gruenstein, A.; and Dowding J. 2003. Targeted help for spoken dialogue systems: intelligent feed back improves naive users' performance. In *Proceedings of European Association for Computational Linguistics (EACL 03)*, (in press).

Lemon, O.; Bracy, A.; Gruenstein, A.; and Peters, S. 2001. Information states in a multi-modal dialogue system for human-robot conversation. In Kühnlein, P.; Reiser, H.; and Zeevat, H., eds., *5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog 2001)*, 57 – 67.

Lemon, O.; Gruenstein, A.; Battle, A.; and Peters, S. 2002. Multi-tasking and collaborative activities in dialogue systems. In *Proceedings of 3rd SIGdial Workshop on Discourse and Dialogue*, 113 – 124.

Lemon, O.; Gruenstein, A.; and Peters, S. 2002. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL)* 43(2):131 – 154. Special Issue on Dialogue.

Nuance. 2003. `http://www.nuance.com/prodserv/prodvocalizer.html`. As of 21 January 2003.

Shapiro, D.; Langley, P.; and Shachter, R. 2001. Using background knowledge to speed reinforcement learning. In *Fifth International Conference on Autonomous Agents*, 254 – 261. ACM Press.

Shieber, S. M.; van Noord, G.; Pereira, F. C. N.; and Moore, R. C. 1990. Semantic-head-driven generation. *Computational Linguistics* 16(1):30–42.

Stent, A. 1999. Content planning and generation in continuous-speech spoken dialog systems. In *Proceedings of KI'99 workshop "May I Speak Freely?"*.

Taylor, P.; Black, A.; and Caley, R. 1998. The architecture of the the festival speech synthesis system. In *Third International Workshop on Speech Synthesis, Sydney, Australia*.

Traum, D. R., and Allen, J. 1994. Discourse obligations in dialogue processing. In *Proceedings of the Thirty-Second Annual Meeting of the Association for Computational Linguistics*, 1–8.

Zinn, C.; Moore, J. D.; and Core, M. G. 2002. A 3-tier planning architecture for managing tutorial dialogue,. In *Intelligent Tutoring Systems, Sixth International Conference (ITS 2002)*.