DIALOGUE MANAGEMENT IN A MIXED-INITIATIVE, COOPERATIVE, SPOKEN LANGUAGE SYSTEM

Sandra Williams

BT Laboratories
Martlesham Heath
Ipswich IP5 7RE U.K.
swilliams@info.bt.co.uk.

ABSTRACT

This paper describes a dialogue manager for a spoken language system which allows people to access their email by making a telephone call. The dialogue manager accomplishes three tasks: 1) it controls interactions between the Natural Language Processing (NLP) components; 2) it coordinates inputs from the speech recogniser, requests to and from the database query module and outputs to the text-to-speech module; and 3) it dynamically builds a model of the conversational structure and interprets the dialogue at a higher level than individual utterances. The model of conversation enables mixed initiative dialogues where either the caller or the system can take control. Callers can converse with the system, asking it to search for and read out email messages and to perform a number of common email tasks such as forwarding messages, replying to messages, deleting messages, and filing messages in folders. Anaphoric references and ellipses are resolved by referring to the component of the conversational model which records the dialogue history.

When the dialogue manager cannot find exactly what a user is looking for, it is able to send out a cooperative reply. This reply will give the user some extra information which we hope will be more helpful than just saying "none found".

1. INTRODUCTION

This paper introduces a dialogue manager for a spoken language system, MailSec, currently under development at BT Labs. MailSec allows people to access their email by making a telephone call [1]. An overview of MailSec is given in part 2. The dialogue manager is part of the NLP component of MailSec which is summarised in part 3.

Automatic access to information by telephone has been available since the mid 1980s. Earlier systems had touch-tone input only, then isolated word speech recognisers were developed that were accurate enough to allow callers to say numbers (digit by digit) and a few keywords (`yes', `no', and so on). Current commercial systems can handle much larger vocabularies and speaker-independent speech recognisers can cope with a mixture of isolated words, connected digits, and connected alphabetical letters. An example of this is an automated directory enquiry system currently in use at BT Labs. [2] which holds a database of several thousand people's names and telephone numbers.

In the future, conversations between humans and machines will be far more natural than they are at present. We foresee that they will become more like human-human conversations but the machine will be a more attentive listener than a human; it will have more consistent reasoning skills than a human; and it will be better and more accurately informed than a human.

MailSec's dialogue manager, described in this paper, goes some of the way towards achieving this. It enables more natural dialogues between the caller and system than has been possible to date. One of the ways MailSec does this is to allow either the caller or MailSec to take the initiative in steering the course of the dialogue (see part 5). MailSec attempts to `understand' more than individual utterances in isolation. It attempts to `understand' the entire conversation by building a conversational model (see part 4). This means it is capable of resolving between-turn references (see part 6) and of leaving one request and answering another before returning to the original request. MailSec is able to provide co-operative responses (see part 7) when requested information is not there. This means that it tries to provide some information which might be useful, rather than just saying `none found'.

2. OVERVIEW OF THE TELEPHONE EMAIL SYSTEM: MAILSEC

MailSec is designed for use by people who are away from their offices and who do not have access to a computer and modem, people who nevertheless want to have access to their email. MailSec enables them to ring up and have their emails listed or read out to them. In addition to this, emails can be forwarded, deleted, replied to, and filed.

The application is similar in functionality to the email part of Sun Microsystems 's Speech Acts [3], but our system allows callers to converse in a more natural manner and it incorporates discourse features, such as reference resolution, which SpeechActs does not. Part of a typical conversation between a caller and MailSec might progress as follows:

Caller:	List the new messages.
MailSec:	You have two new messages, one from Esther Vennell entitled 'Stationery', sent on 20th May and one from Keith Preston entitled 'Visit', sent on 20th May.
Caller:	Delete the one from Esther.
MailSec:	Delete the message from Esther Vennell entitled 'Stationery'?
Caller:	Yes.
MailSec:	OK, done.
Caller:	Read Keith's email.
MailSec:	Message from Keith Preston entitled 'Visit', message reads '.....'.

Spoken Language Systems combine the three technologies of Speech Recognition, Natural Language Processing (NLP) and text-to-speech as shown in Figure 1.

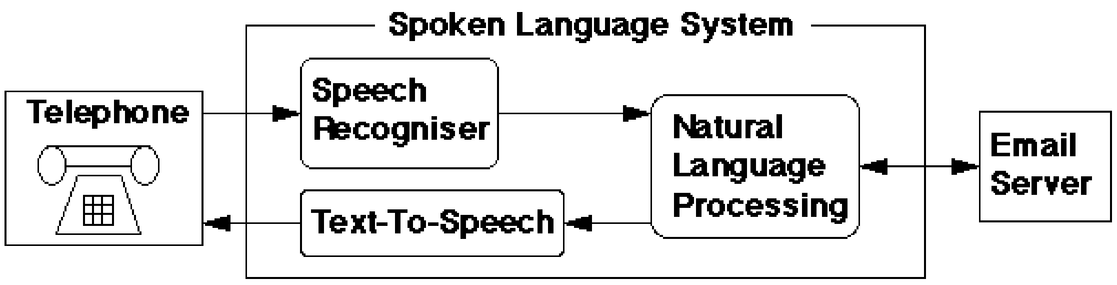

Figure 1. Schematic diagram showing MailSec's major technologies

Figure 1 is a schematic diagram to show the technologies involved. It does not reflect the system architecture. For this see [1]. The figure has been simplified and many details such as the telephony interface software, interfaces to various different email servers, and interfaces between the Spoken Language System components themselves have not been shown.

The speech recognition component of MailSec is the BT Labs continuous speech recogniser [4] which allows users to speak in a natural manner without having to leave pauses between words. It is a speaker-independent recogniser which means that it can recognise a wide variety of English speakers regardless of accent or voice pitch and modulation. This has obvious advantages over a speaker-dependent recogniser which would have to be trained on all callers' voices before it would be able to recognise them. The function of the speech recogniser is to recognise the incoming speech, and transcribe it from a speech waveform into a textual representation.

For outgoing speech, MailSec uses the BT Labs text-to-speech (TTS) system, Laureate [5]. Laureate's speech is derived from a real human voice and is one of the most natural-sounding TTS systems currently available. Laureate converts a string of outgoing text into a speech waveform to be played back the caller.

The NLP part of the system interprets the meaning of the incoming speech from the caller which has been transcribed into text by the speech recogniser. It collects information from an email server, and constructs the replies as text for Laureate to convert to a speech waveform. This paper is concerned with the dialogue manager which controls interactions between the natural language processing (NLP) components; coordinates inputs from the speech recogniser, requests to and from the database query module and outputs to the text-to-speech module whilst dynamically building a model of the conversational structure.

3. OVERVIEW OF THE NLP COMPONENT

The dialogue manager is part of the NLP component of MailSec. Other modules include: a parsing/semantics module, an anaphoric reference and ellipsis resolution module, a database query module, a cooperative response module, and a text generation module. These modules, together with the data structures built by the dialogue manager are shown below in figure 2.

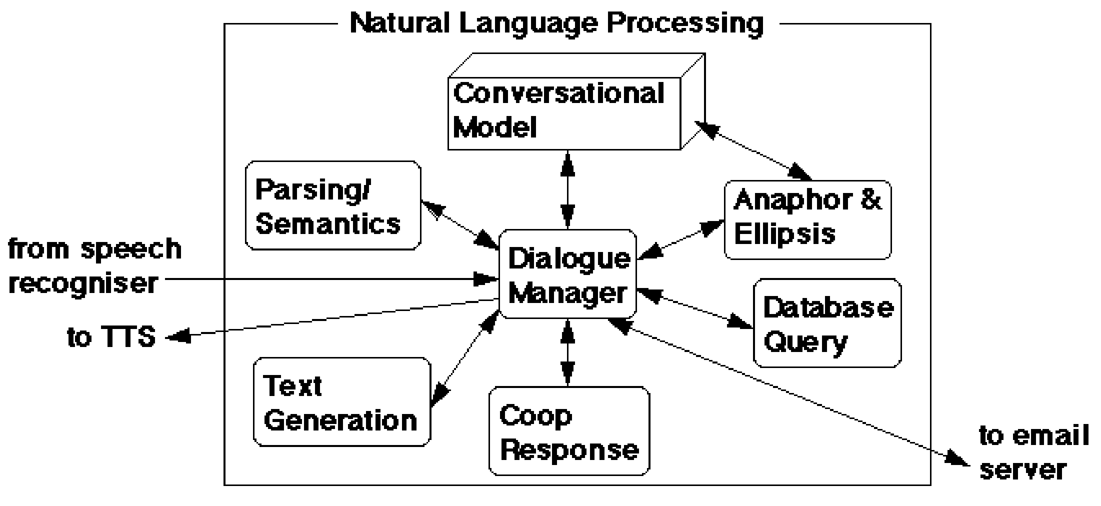

Figure 2. Schematic diagram of MailSec's NLP component

Figure 2 is a schematic diagram showing how the dialogue manager relates to the other NLP modules. It leaves out many details of the individual modules and those data sources used by modules other than the dialogue manager and the anaphoric reference and ellipsis resolution module. For a more detailed description of the architecture, see [1].

The dialogue manager controls the flow of processing of data by the other NLP modules and coordinates data flowing to/from the speech recogniser, email server, and TTS. A typical processing sequence proceeds in a clockwise direction in figure 2 from the speech recogniser input, through the parsing/semantics module, then to anaphor/ellipsis resolution, then to database query, then to cooperative response analysis, then to text generation and finally out to text-to-speech. The conversational model is modified and consulted at various stages throughout the processing. Note however that the NLP modules may be called by the dialogue manager in a flexible way. An example of this is when the dialogue manager uses the cooperative response module to modify the query, and then database query is attempted again. Brief descriptions of each module are given below.

3.1 The Dialogue Manager

The dialogue manager maintains the conversation between MailSec and the caller. It co-ordinates the operation of all other system components and builds a dynamic model of the conversation as it progresses.

Individual utterances cannot be totally meaningful and coherent unless they are interpreted in their correct context within the conversation. If a caller says `Anna', or `Yes', or `Read the next', the semantic representation alone does not give enough information for MailSec to understand and know what to do next. Therefore on receiving the parse for the input utterance, the dialogue manager interprets its meaning in the context of the conversation and builds it in to the conversational model. The model ensures that answers are matched up with corresponding questions, and so on. In this way we build up a representation of the dialogue meaning above the level of individual utterances.

This paper is chiefly concerned with the dialogue manager, its associated data structures, and the functionality it provides.

3.2 The Parsing and Semantics Module

On receiving input from the speech recogniser, the dialogue manager sends it to the parsing and semantics module which interprets the meaning of the caller's utterance. The meaning is represented by what we have called an Extended Logical Form (ELF). The ELF is constructed using the standard PROLOG notation where variables are capitalised, and atoms are lower case. An example of an ELF for 'List any emails from Anna' is:

ELF: list(A),

Expect List: [plural(A)],

Spec List: [message(A), from(A,B), name(B,anna), gender(B,feminine)].

The ELF is a non-recursive form with three parts. The first shows that there exists something, A, that is to be listed. The second is a list of facts indicating what the caller is expecting, the Expect List. In this example, the caller is expecting A to be plural ('emails' in the input string). The Expect List is used by the text generator (together with the database query result and the Spec List) to build an appropriate response. The third part is a specification of what the caller requires ('emails from Anna'). The Expect List and Spec List contain a number of conjoined constraints. In this case, there exists something, A, where A is a message, and A is from B where B is named 'anna' and feminine. The Spec List is used by the dialogue manager as the base for building the database query.

3.3 Conversational Modelling

Our model of the conversation is built dynamically as the conversation progresses and it is based on the theory of Games Structure in Conversation devised by Kowtko et al [6]. The model is our own interpretation of the theory of Games Structure in conversation. Other systems have also successfully made use of this theory: [7] and [8]. The model has two parts:]

* The History List which contains all previous utterances. This is used to resolve anaphoric references and ellipsis and for requests from the caller to `Repeat'.

* The stack of conversational games. This is used by the dialogue manager to track the conversation and 'understand' the significance of an utterance in the context of the dialogue and work out how to respond to the caller.

For more details, see part 4.

3.4 Anaphoric Reference and Ellipsis Resolution Module

This is the module that resolves ellipsis and anaphoric references. Anaphoric references occur when a speaker refers to something mentioned earlier in the dialogue (e.g. 'his' in 'read his email'). Ellipsis occurs when something is missing from a speaker's utterance which can be understood from the context (e.g. 'emails' in 'How many ... from Anna do I have?').

Not all incoming utterances undergo reference resolution. For instance if MailSec has just asked a yes-no question (e.g. `Do you want to delete the message from Alison Simons entitled "Meeting"') and the caller just answers 'yes' or 'no', then this module is bypassed.

The ellipsis and reference resolution module is described in detail below in part 6.

3.5 Database Query Module

The database query module is the part of the system that searches the email database for emails that match the query supplied by the dialogue manager.

3.6 Cooperative Response Module

If the database query does not find anything, the cooperative response module is used to modify the query in order that a cooperative response might be given to the caller. See part 7 for more details.

3.7 Text Generation Module

The text generation module constructs the text for the system's response to the caller.

4. CONVERSATIONAL MODEL

The conversational model consists of two parts: the conversational games stack and the history list. These are described in detail below in 4.1 and 4.2.

4.1 Conversational games stack

The dialogue manager implements the theory of `Games Structure in Conversation' [6]. According to this theory, conversations can be analysed as a series of `games' and `moves', each one denoted by a game beginning (such as the INSTRUCT move: `Tell me the time.') and a game end (such as the ACKNOWLEDGE move: `OK, Thanks.'). There can be many, or few, turns in the conversation between a game beginning and a game end. Because of the way games tend to be nested, the overall control structure can be represented as a last-in-first-out push-down stack. This Conversational Games Stack, together with the History List, form the core data structure of the dialogue manager.

Kowtko et al have defined six types of Conversational Games (INSTRUCT, CHECK, QUERY-YN, QUERY-W, EXPLAIN, ALIGN) and 12 kinds of game move. The moves are summarised in tables 1 and 2 below:

Instruct	Communicates a direct or indirect request or instruction, to be done immediately or shortly. e.g. "You then go down two inches."
Check	.Checks self-understanding of a previous message or instruction by requesting confirmation directly or indirectly; makes sure that a complicated instruction is understood. e.g. "So you want me to go down two inches?"
Query-YN	Yes-No question (QUERY-YN) or open-answer Wh-question (QUERY-W)
Query-W	asks for new or unknown detail about some part of task; does not request clarification about instructions (that would be a CHECK); e.g. "Do you have a rockfall?"
Explain	Describes status quo or position in task with respect to the goal; freely offered, not elicited; provides new information. e.g. "I've got a cairn."
Align	Checks the other participant's understanding or accomplishment of a goal; elicits a positive response which closes a larger game; checks alignment of both participants' plans or positions in task with respect to goal; checks attention, agreement, or readiness. e.g. "Ok?", meaning, "Are you with me?"

Table 1: Opening Moves [6]

Clarify	Clarifies or rephrases what has previously been said; usually repeats given or known information; elicited by other person. e.g. "South, two inches."
Reply-Y	Affirmative (REPLY-Y) or negative (REPLY-N), elicited response
Reply-N	to QUERY-YN, CHECK, or ALIGN; also indicates agreement, disagreement, or denial; e.g. "Yes, I have."
Reply-W	An elicited reply to QUERY-W or CHECK; can be a response to QUERY-YN that is not easily categorizeable as positive or negative (REPLY-Y/N). e.g. "Down"
Acknowledg e	Vocal acknowledgement of having heard and understood; not specifically elicited but often expected before the other speaker will continue; announces readiness to hear next move - in essence a request to `please continue'; may close a game. e.g. "All right"
Ready	Indicates intention to begin a new game and focuses attention on oneself, in preparation for the new move; an acknowledgement that the previous game has

	just been completed, or leaving the previous level or game; consists of a cue-word. e.g. "Now" or "Right."

Table 2: Other Moves [6]

The six games correspond with the six kinds of opening move in Table 1. Therefore an Instruct game will always start with an INSTRUCT move, a Check game with a CHECK move, and so on. the READY move is a kind of pre-game signaller, so in effect some games will actually begin with READY.

A game may typically consist of a simple pair of moves, for instance a question (e.g. QUERY-YN) and an answer (e.g. REPLY-N). However, a game may consists of only one move, or it may have more than two moves. For instance, an Explain game may consist of just one EXPLAIN move (e.g. at the beginning of the call, MailSec tells the caller how many new messages there are), but a Query-W game which begins with a QUERY-W, may not end with a REPLY-W, there may be an additional ACKNOWLEDGE move such as "OK" or "Thank you". In practice, the system ends the game after the REPLY-W move and any spurious ACKNOWLEDGE moves are thrown away.

A typical conversation between MailSec and a caller (recorded at BT Labs) is analysed in Figure 3.

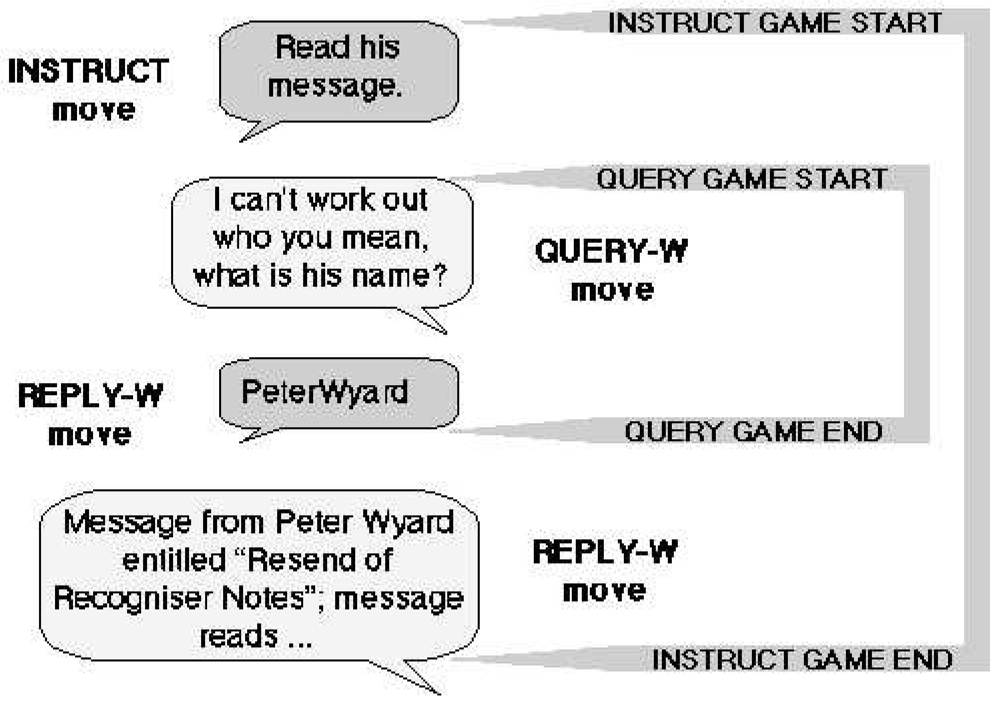

Figure 3: Analysis of a dialogue using Games Structure in Conversation [6]

The first talk bubble is the caller speaking: `Read his message'. This is an INSTRUCT move (an imperative) and it starts an INSTRUCT game. The dialogue manager pushes an INSTRUCT game started by the caller onto the games stack.

The second talk bubble is MailSec's reply: `I can't work out who you mean ...'. This is a QUERY-W move (a WH question) the start of a QUERY-W game. The dialogue manager pushes a QUERY-W game started by MailSec onto the games stack.

The third bubble is the caller saying `Peter Wyard'. This is a REPLY-W move and it completes the QUERY-W game. The dialogue manager pops MailSec's QUERY-W game from the game stack.

The final talk bubble is MailSec's REPLY-W move reading out the email and ending the INSTRUCT game. The dialogue manager pops the caller's INSTRUCT game from the game stack.

Note the nested nature of the games. Note also that:

* A turn in the conversation may contain more than one move. This has not been implemented yet.

* A move can end more than one game. An example of this is when a caller is answering a question (e.g. a QUERY-YN) and the speech recogniser does not recognise the input, so MailSec says "Pardon?" (QUERY-W). In repeating the utterance, the caller is both ending the QUERY-YN game as well as the new QUERY-W game.

Our implementation of this theory enables mixed initiative dialogues described in section 5. When the caller's utterance first arrives from the speech recogniser, the dialogue manager builds it into the conversational model. If MailSec has just asked a question, then the dialogue manager attempts first to process the incoming utterance as a reply to the question. If this fails, then it is processed as the start of a new game (usually an INSTRUCT game, a QUERY-W game, or a QUERY-YN game). The output from the parsing and semantics module is used to determine which new game is starting.

If the caller begins a new game unexpectedly, then MailSec's question is retained on the stack so that it can check again with the caller later. For instance, when MailSec asked for the name in the above dialogue, the user could have said `How many new emails do I have?' This would start a new QUERY-W game on the third level of the stack. MailSec would answer this question immediately, then pop the game off the stack. Seeing two levels already on the stack, it would then ask the user again `Do you still want to read the message from him?'. If the caller chose to ignore the question again, in our current implementation, MailSec will pop these two levels from the stack and forget them.

The dialogue manager's behaviour can be altered to fit in with the requirements of different applications. For some applications certain pieces of information might be essential and it might be necessary for the caller to supply them in a particular order. In cases like this, the dialogue manager's behaviour could be altered so that unanswered questions do not disappear from the conversational games stack. There is, of course, a limit to the number of times a caller will tolerate being badgered with the same question time after time, even if the words are varied, before he/she gives up.

4.2 History List

The History List records every utterance generated by the caller, together with the corresponding response by MailSec. The response from MailSec is recorded in the History List in the same entry as the caller's utterance. This may be the result of a database query, or it may be a question MailSec asked the user, e.g. QUERY-W(unknown_sender) for questioning the name of the sender of an email.

The History List is used to resolve ellipsis and anaphoric references (see part 6), and also to respond to user requests to `Repeat '.

5. MIXED INITIATIVE DIALOGUES

Many existing commercial telephone dialogue systems use prompt and response. The system plays out a prompt and the caller has to speak after the prompt and give the information required. MailSec takes a different approach. We are looking beyond prompt and response and aim at mixed-initiative dialogues where the caller does not necessarily have to answer the system's questions but can choose to ignore them and give some other command or ask the system a question back. Our view of mixed initiative in conversation corresponds with that of Walker and Whittaker [9], but our implementation is not rule-based like theirs.

To demonstrate what we mean, we give examples of these two different types of dialogue below.

First we show an example of a prompt and response dialogue between a caller and the large vocabulary system mentioned in the introduction [2]:

System:	Welcome to the BT Corporate Directory Enquiry System. Please say the surname.
Caller:	Williams
System:	And now please SPELL the surname.
Caller:	W-I-double-L-I-A-M-S
System:	Thank you. Please say the first name.
Caller:	Sandra
System:	Is the name Sandra Williams?
Caller:	Yes
System:	The number is 01473 605660. I'll repeat that. The number is 01473 605660. If you would like to make another enquiry please hold on, otherwise please replace the handset. Thank you for calling.

This dialogue is controlled by the system which means that the system takes all the initiative in the conversation. The caller cannot say anything other than what has been requested. If the system asks a question, it will insist that the question be answered before the dialogue can progress.

Prompt and response style dialogues are very useful in systems like this that give out very limited and specific information (such as a telephone number) and require very limited and specific information from the caller (e.g. a first name and a surname) in order to do so.

Our system is required to be more flexible than this, and can perform a number of tasks as well as give out information. Since the caller will probably only decide on the next thing to do after receiving some information, it is much better if the caller can ask MailSec directly what he/she wants to do next at the point when he/she decides what this is. The alternative to our flexible dialogue approach would be for the caller to navigate through a long and tedious series of menus before finding the information or task he/she wants. This would break the flow of the conversation, take longer, and interrupt the caller's train of thought. Below is an example of a mixed initiative conversation:

Caller:	Read the messages from Alison
MailSec:	You have three messages from Alison Simons. Message from Alison Simons entitled "RE: MailSec" , message reads "I agree with Peter's comments about MailSec". Do you want to read the next message?
Caller:	Read Peter's email about MailSec.
MailSec:	Message from Peter Wyard entitled "'MailSec", message reads "At our last meeting we discussed ...". Do you still want to read the messages from Alison Simons?

The caller asks MailSec to read Alison's messages. MailSec reads the first and asks if the caller wants to read the next (at this point MailSec has control). The caller chooses to ignore this question and asks to read Peter's email that Alison mentioned (now the caller takes control). MailSec reads the message and asks again whether to read messages from Alison Simons. If the caller ignores the question again, then MailSec will not ask again until the caller says 'goodbye', when it will do so as a final reminder. Obviously we can change the behaviour so that MailSec will ask as many times as we program it to, but we do not want the dialogue to become tedious and we want to ensure that the caller is always free to do as he/she likes.

6. REFERENCE AND ELLIPSIS RESOLUTION

Reference and ellipsis are common discourse phenomena in natural dialogues. The reference and ellipsis resolution module is separate from the dialogue manager, but since the resolution process is so dependent on the History List part of the Conversational Model, we include a brief description of our treatment here for completeness, but for more details see [10].

When speaking to MailSec, the caller can refer in a comfortable and natural way to an email mentioned earlier in the conversation. This is illustrated by 'it' in 'Read it' in the following dialogue:

Caller:	Do I have any messages from Anna?
MailSec:	You have one new message from Anna Cordon entitled 'MT Meeting'.
Caller:	Read it.

6.1 Reference resolution

Anaphoric references are a specific kind of referring expression and are used when a speaker refers back to something mentioned earlier in the discourse, e.g. "he", "him", "her", "that one", "that way", "the house" can all be anaphoric reference expressions.

Caller:	Are there any messages from Peter?
MailSec:	You have two messages from Peter Wyard, one is entitled `Meeting on Tues' and one is entitled `BusCat Demo'.
Caller:	Read his second email.

In the above exchange, the caller's second utterance contains the anaphoric reference `his'. The parsing/semantic module represents "Read his second email." as an Extended Logical Form

(ELF - see part 3.2):

ELF: read(A),

Expect List: [salient(A),singular(A)],

Spec List: [message(A), ord(A,2), named(B,C),gender(B,masculine),from(A,B)]

It is the reference resolution process which fills in unknown information in the ELF. It does this by searching back through the preceding logical forms to find a masculine person. We are assuming that the last masculine person mentioned will be the referent of `his', although this might not always be the case. The ELF output by the parsing and semantics module is thus converted into a Resolved Extended Logical Form (RELF) where named(B,C) is changed to named(B,peter):

RELF: read(A),

Expect List: [salient(A),singular(A)],

Spec List: [message(A), from(A,B), ord(A,2), named(B,peter),gender(B,masculine)]

6.2 Ellipsis resolution

Ellipsis occurs when something is left out of an utterance which can be determined from what has gone before:

Caller:	Do I have any messages from Peter?
MailSec:	You have two messages from Peter Wyard.
Caller:	Do I have any from David?

The caller has left out the noun `messages' and has simply said `any'. For this utterance, the parsing/semantics module will produce the extended logical form:

ELF: A,

Expect List: [],

Spec List: [from(A,B), named(B,david), gender(B,masculine)]

The ellipsis resolution module searches the Spec List for an entity relevant to the domain. In the email domain, entities can be emails, folders, or headers. If there is no entity present, then it searches for the missing information in previous logical forms in the History List. It looks for the most recently mentioned entity, `messages'. It adds this to the Spec List of the ELF to produce the following RELF:

RELF: list(A),

Expect List: [],

Spec List: [message(A), from(A,B), named(B,david), gender(B,masculine)]

7. COOPERATIVE RESPONSES

We are experimenting with cooperative responses for situations when the database query module finds no matching emails. The cooperative response module is separate from the dialogue manager, but it serves as a good example of how the dialogue manager monitors the results from other NLP modules and directs the flow of processing accordingly. If the database query module finds nothing, then rather than just saying `none found', the dialogue manager attempts to give some information which the user might find helpful. It uses the cooperative response module to generalise the query, without making it so general that any email would match. If some emails match a more general query, then the dialogue manager arranges for a cooperative response to be sent to the user.

Caller:	Do I have any emails from Anna about elephants?
MailSec:	No, you have no emails from Anna Cordon about elephants, but you have one message from Keith Preston about elephants.
Caller:	OK, but do I have any emails at all from Anna?

Here the query has been generalised by dropping the condition `from Anna' and MailSec has tried to be helpful by looking for other messages about elephants.

In order to generalise the query, the dialogue manager must decide which parts of the user's specification are the most important. Is it more important that the email is from Anna or that the email is about elephants? Or are the two equally important? Here MailSec wrongly assumes that the user is primarily interested in elephants. This whole area of cooperativeness is one which we intend to investigate further in the future.

When the database query is generalised with a successful result, the dialogue manager produced another logical form, the Cooperative Extended Logical Form (CoopELF). For the above example, the RELF and CoopELF are as follows:

RELF: list(A),

Expect List: [plural(A)]

Spec List: [message(A), from(A,B), name(B,anna), gender(B,feminine), about(A,elephants)]

CoopELF: list(A),

Expect List: [plural(A)]

Spec List: [message(A), about(A,elephants)]

Both logical forms are used by the text generation module in constructing the response.

8. CONCLUSIONS

We have successfully implemented an experimental dialogue manager for a telephone-based email spoken language system which allows more natural dialogues than existing commercial systems. It enables a mixed initiative dialogue by using a model of `Games Structure in Conversation' [6]. It resolves between-utterance references and ellipsis. It provides helpful cooperative responses for the caller.

As it develops, the dialogue manager implementation is improving and becoming more robust. In the future it will become more domain independent when we gain experience from porting it to other domains.

In building this dialogue manager we have made a significant step forward towards more natural man-machine dialogues.

ACKNOWLEDGEMENTS

We are grateful to the members of the BT Labs NLP Group, and members of the BT Labs Spoken Language System development team for their help, encouragement and advice during this work. In particular we would like to thank David Franklin, Louise Helliker, Christopher Holt and Alison Simons. Special thanks to Peter Wyard, Edward Kaneen and Stephen Appleby for their work on the parsing and semantics module and to Keith Preston for his work on the database query module.

References

1.. Wyard,P.J. Simons,A.D. Appleby,S. Kaneen,E. Williams,S.H. and Preston,K.R. `Spoken Language Systems - beyond prompt and response', BT Technology Journal, Vol. 14, No. 1, January 1996.

2. Attwater, D.J. and Whittaker, S.J. `Issues in large vocabulary interactive speech recognition', BT Technology Journal, Vol. 14, No. 1, January 1996.

3. Yankelovich, N. and Baatz, E. 'SpeechActs: a framework for building speech applications', AVIOS `94 Conference Proceedings, San Jose, CA, September 20-23, 1994. SMLI 94-0243.

4. Scahill,F. Talintyre,J.E. Johnson,S.H. Bass,A.E. Lear,J.A. Franklin,D.J. and Lee,P.R. `Speech Recognition - making it work', BT Technology Journal, Vol. 14, No. 1, January 1996.

5. Page,J.H. and Breen,A.P. `The Laureate text-to-speech system - architecture and applications', BT Technology Journal, Vol. 14, No. 1, January 1996.

6. Kowtko, J.C. Isard, S.D. and Doherty, G.M., `Conversational Games Within Dialogue', Human Communication Research Centre, Edinburgh University, 1994

7. Lewin,I. Russell,M. Carter,D. Browning,S. Ponting,K. and Pulman,S. `A speech-based route enquiry system built from general-purpose components', Eurospeech'93, 3rd Conference on Speech Communication and Technology, Berlin, 1993.

8. Bird,S. Browning,S. Moore,R. and Russell,M. `Dialogue move recognition using topic spotting techniques', ESCA Workshop on Spoken Dialogue Systems, Denmark, 1995.

9. Walker,M. and Whittaker,S. `Mixed initiative in dialogue: an investigation into discourse segmentation', ACL Pittsburgh, 1992.

10. Williams,S.H. (forthcoming) `Anaphoric reference and ellipsis resolution in a telephone-based spoken language system for accessing email', DAARC'96, 1996.