Simulation of human-machine dialogues

K.H. Scheffler & S.J. Young

CUED/F-INFENG/TR 355

September 1999

Speech, Vision and Robotics Group
Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

E-mail: khs22,sjy@eng.cam.ac.uk

Abstract

The field of spoken dialogue systems has developed rapidly in recent years. However, optimisation, evaluation and rapid development of systems remain problematic. In this research, a method was developed to produce probabilistic simulations of mixed initiative dialogue with recognition and understanding errors. Both user behaviour and system errors are modelled using a data-driven approach, and the quality of the simulations are evaluated by comparing them to real human-machine dialogues.

The simulation system can be used to perform rapid evaluations of prototype systems, thus aiding the development process. It is also envisaged that it will be used as a tool for automation of dialogue design.

Contents

1 Introduction

The field of spoken dialogue systems has developed rapidly in recent years. The goal of this field is to develop computer systems with which humans can interact as if in a natural conversation, in order to perform tasks such as information retrieval. While a small number of such systems have been deployed, optimisation and rapid development of systems remain problematic.
In this research, a probabilistic method of dialogue simulation was developed to aid the process of developing new dialogue systems. It is also envisaged that the simulation system will be used as a tool for automation of the design process.

1.1 Basic dialogue system architecture

The basic components found in a typical spoken dialogue system (SDS) are illustrated in Figure 1. The diagram depicts the flow of information from the human user, to the system and back again. The essential components are subsystems for input (conveying information from the user to the system), dialogue management (deciding how to react) and output (conveying information from the system back to the user).

System input: Spoken dialogue systems typically make use of Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) systems to convert the input speech signal first to word strings and then to convenient abstract representations of information. Such systems are commercially available and can be thought of as separate black boxes.

Dialogue management: The central component of a dialogue system is the dialogue manager. This is the component that coordinates the flow of the dialogue by deciding on the actions to be taken by the system. It also manages interaction with the external world through utilities such as databases or other objects the computer has access to. In this research, we consider dialogue managers that can be described as finite state automata, with states that correspond to the dialogue circumstances. Such systems implement a dialogue strategy that maps states into actions.

System output: The final requirement of a dialogue system is some way of communicating information back to the user. This can again be done by means of speech, using subsystems for natural language generation and text-to-speech synthesis. Natural language generation is typically performed by simply using fixed phrases chosen for the situations that may occur, while production of speech may also be done by using recorded speech segments.

The information transferred between the various subsystems of figure 1 can be interpreted at the following levels of interaction [1]:

Speech level: Speech is the signal that is physically transmitted from the user to the system. Thus the speech level is the most obvious interface between the dialogue participants.

Word level: One could also think of the speech recognition and text-to-speech units as part of the transmission channel between the participants. In this

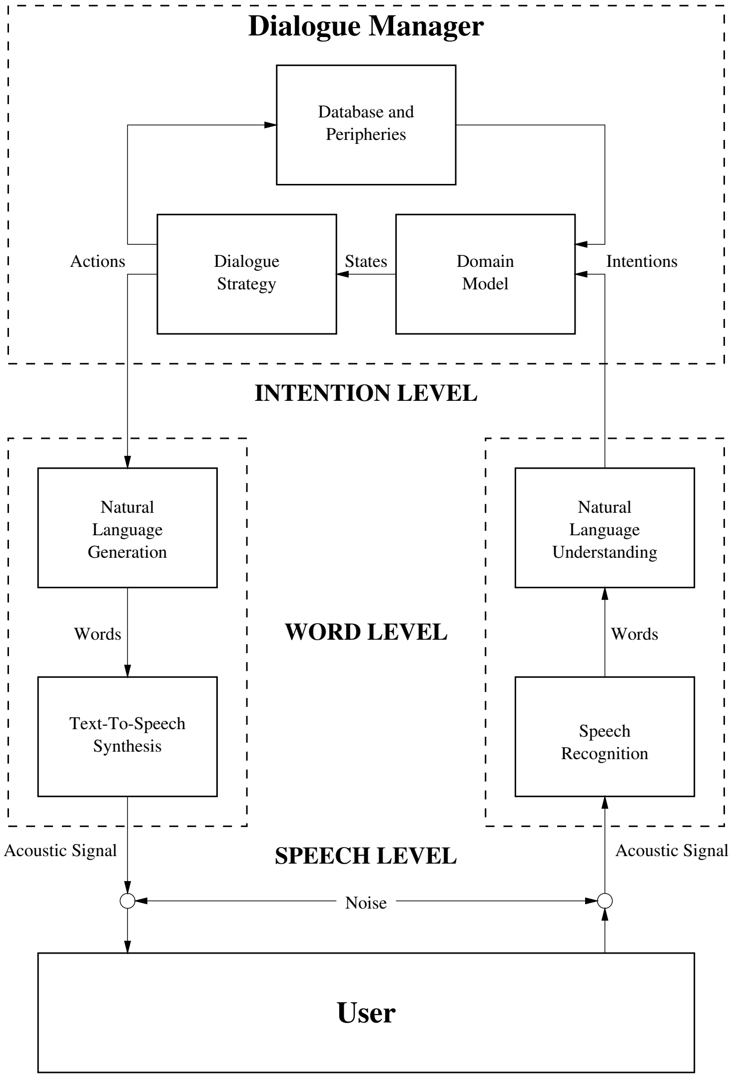

Figure 1: Block diagram of a typical spoken dialogue system.

view, the interface is at the word level, with communication taking place in a text format. Note that the distortion in the transmission channel is increased in this perspective, as it now includes speech recognition errors.

Intention level: Taking the idea of a word level interface one step further, we arrive at the intention level. Intentions represent the actual information (concepts) that a dialogue participant wants to convey, and are analogous to the speech acts or dialogue acts of classical dialogue theory. All interpretation of the physical signals is now considered part of the communication channel, allowing us to think of intentions as being conveyed directly, and highlighting the dialogue manager as the core of the dialogue system. An intention can be defined as the minimum piece of information that can be conveyed independently within a given application.

1.2 Major problems in the field of dialogue systems

A number of problems in the field of spoken dialogue systems are regarded as major obstacles or open questions. These include:

Portability: Although systems have been implemented for many specific applications, it remains a huge task to port a system to a new application domain. The design of the dialogue manager is a particular problem in this respect, since every new task presents new, unforeseen situations where the dialogue manager has to decide what to do.

Optimisation: It is desirable to design systems so that they are optimal with respect to some criterion. In the case of SDS, commonly used criteria are dialogue success rate and average dialogue length. However, it is by no means obvious how to optimise a system using these criteria.

Evaluation: There is no standard method for evaluating a dialogue system or even comparing the performance of different systems. It is impossible, for instance, to feed a standard corpus of utterances to a dialogue system: a dialogue is the result of an interaction between the user and the system, not just of the system reacting to isolated user utterances. Thus a full system evaluation can only be done by letting the system interact with real users, which is of course expensive and time consuming.

An alternative approach for evaluating systems is to create a model of the users they will interact with. Dialogues can then be simulated and performance measured on the simulated rather than real dialogues [2]. If the user model, and thus the simulation, is realistic, this will yield an adequate evaluation of the system that can be performed without the expense and effort of running tests with real humans. Also, this kind of evaluation can be repeated many times during the development process, helping developers to optimise various system aspects and generally speeding up the development process. This is of paramount importance if the goal of easy cross-domain portability is to be realised.
A further possible step would be to automate the high level design process completely by using machine learning techniques [3] to find good dialogue strategies

[4]. A system that can simulate dialogues is an important requirement if this approach is to be realised.
This report describes a system that has been developed to enable simulation of human-machine dialogues for the purposes set out above.

1.3 Report outline

The report is organised as follows:
Section 2 introduces the problem to be tackled along with the approach taken. Some central concepts relating to dialogue and utterance structures are also introduced. The first step in implementing a simulation system was to use a simple bigram user model based on the work of Eckert, Levin and Pieraccini [2]. Section 3 describes this work, and argues that a more sophisticated approach is required. Section 4 then develops a goal directed user model to counter the weaknesses of the bigram model. Finally, the simulation system is evaluated and results are given in section 5.

2 Dialogue simulation by probabilistic user modelling

In this section, we describe a system that has been developed to simulate co-operative, task oriented human-machine dialogues, in order to serve as a tool for design and development of dialogue systems. The simulation system was developed in the context of a specific application in the banking domain, from which the examples will be taken. However, the principles on which the system is based are domain independent so that porting to different domains should not present serious problems.
The main focus of the simulations is to enable evaluation of a speech based dialogue system. For this to be possible, it is necessary to model both the users with whom the system interacts, and the speech recognition/understanding errors made by the system.

2.1 Problem statement

The problem statement for the simulation includes the following:

2.1.1 Dialogue system

- A prototype system is available and can interact with humans.
- The system is deterministic, always reacting to the same situation in the same way. The number of such situations is finite, as is the number of actions that can be taken by the system.
- The system accepts natural speech input.
- The system supports mixed initiative.
- The application is complex enough to require both loops and confirmation subdialogues in the dialogue structure.

2.1.2 Human user

- The conversation partner for the system is a human user, for whom an "implementation" is not available.[1]
- The user acts in an unpredictable, non-deterministic way, as seen from the viewpoint of the system.[2]

2.1.3 Communication channel

- The dialogue is conducted via a "noisy" communication channel. In this context, what we regard as noise includes not only speech recognition errors, but also understanding errors as well as errors on the part of the human user in formulating his/her intentions.

For the purpose of this research, a prototype dialogue system was used. The system provides a telephonic, speech based interface for a banking application, supporting transfer transactions between accounts, enquiries of the caller's account balances and stock quote enquiries. The system is implemented using a finite state dialogue structure. The allowable syntax for speech input is defined for each dialogue state, by means of a finite state word network. The word networks are designed so as to support natural language, mixed initiative speech input. In this report, this particular system will be referred to as the *banking application*.

2.2 Simulation methodology

The components needed to implement a dialogue simulation correspond to the problem aspects highlighted in section 2.1: the prototype dialogue system has to be interfaced with a model of the user, and some form of error modelling has to be incorporated.

2.2.1 Interface level

The interface between the prototype system and the user model has to use a level of interaction such as those introduced in section 1.1. The speech and word levels are relatively application independent, but the intentions used on the intention level depend largely on the application and the system implementation. In the banking application, intentions can be mapped to semantic tags which are extracted during the speech recognition process. The following example should clarify what is meant:
Consider the semantic tag "transfer". This could correspond to any of a large number of possible word strings, such as (in the context of an open "How may I help you?" prompt):

- Transfer.
- I'd like to transfer some money.
- 200 pounds to my checking account, please.

---
[1] This is in contrast to the otherwise similar work by Araki and Doshita [5].
[2] That is, the user's behaviour is too complicated to be modelled deterministically.

In the latter case, the “transfer” tag is inferred in spite of the absence of “transfer” or any synonym in the utterance, and occurs along with the tags “cash amount (200)”, “to” and “checking account”. These tags are mapped to the intentions “transfer”, “cash amount (200)” and “to the checking account”.

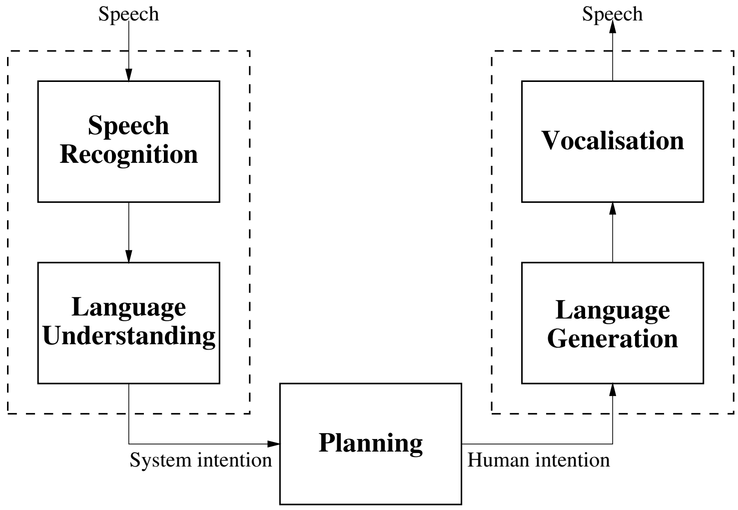

Figure 2: Conceptual block diagram of a human user.

In order to decide at which level of interaction the simulation should be done, we shall think of the user as having hypothetical subcomponents mirroring that of an artificial dialogue system, as shown in Figure 2. If the simulation is to be performed at the speech level, it would be necessary to construct models for not only the central planning component of the user, but also the periphery input and output components. Similarly, a word level simulation would necessitate modelling of the human understanding and language generation processes. The latter is especially problematic, as it would necessitate mapping each possible user intention to a representative set of utterances. Given the wide range of possibilities for phrasing even a simple intention, this would be a huge obstacle. Fortunately, it is possible to perform the simulation at the intention level, bypassing the lower levels completely. We make the assumption that the user can understand the intentions output by the system and that the user's intentions can be mapped onto the intentions understood by the system. The user's recognition and understanding processes are modelled as the inverse of the system's language and speech generation processes, with any deviation to be incorporated into the user model. The required simulation system then reduces to that shown in Figure 3.

The planning component in Figure 2, along with any distortion (such as user misunderstanding) on the channel from the system to the user, is replaced by the user model component. The transfer function of the channel from the user to the system (user language and speech generation, system speech recognition and system understanding) is replaced by the error generation component.

The following arguments motivate the approach of working solely on the inten-

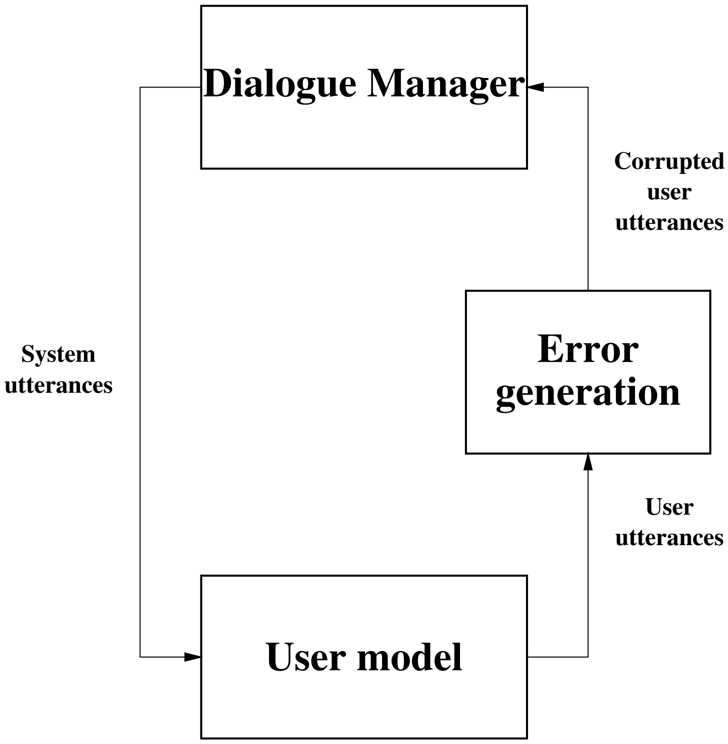

Figure 3: Process of simulating a human-machine dialogue.

tion level:

- Constructing a reasonable model of user behaviour at the word level is probably infeasible, because of the vast domain space of possible user outputs for even a simple utterance. Data sparsity would make the task of parameter estimation impractical if not impossible.
- Modelling user behaviour at the word (or lower) levels is unnecessary for the purpose of testing the high level design of a dialogue system. By incorporating performance statistics from the lower levels (such as natural language understanding), the resulting system ends up simulating the performance of the entire dialogue system anyway.
- By simulating dialogues at the intention level, the results are not restricted to systems of only one language.
- The developed system is equally applicable to non-speech modalities. The only requirement is that errors made by the input system can be modelled.
- By eliminating the linguistic component, the simulation system becomes much more domain independent.

2.2.2 User model

The theory of human-human dialogues distinguishes between two types of model for dialogue agents:

Cognitive models: This type of model is an attempt to describe what is happening inside the mind of the agent. It includes approaches such as belief modelling, viewpoint generation and rational agent models.

Behavioural models: These models describe only the actions of the agents, without incorporating the cognitive processes involved.

While a cognitive model may potentially be useful for constructing a more accurate behaviour model, only a behavioural model is required for dialogue simulation: we are concerned with reproducing reasonable user behaviour but do not need to understand the complex processes involved in causing behaviour.
Another important requirement of the model is that it must be application independent as far as possible. Thus the internal structure of the model should not refer to aspects of the application domain. Rather, it should provide a format in which the necessary information about such aspects can be supplied, and an automatic means of estimating any domain specific parameters.
The user model of Figure 3 is based on the idea that, from the point of view of the system, the user actions appear as the result of an unknown random process that can be modelled probabilistically. Using this approach, it is possible to make the model data driven. Once the structure of a new domain has been defined, it will be possible to estimate those parameters that are domain specific automatically from a domain specific data set.
However, it will be argued in section 3 that a purely probabilistic model is insufficient for dialogues with a realistic level of complexity. This is because real users do not act randomly. Rather, they base their actions on some hidden but deterministic goal, so that different actions in the same dialogue tend to be

consistent with one another. This idea is well known in the theory of human-human dialogues, and the user model developed here has similarities with the classic plan based approach to dialogue modelling.
A major problem in plan based approaches is that of inferring user goals from data. Since we are concerned with generating rather than interpreting dialogues, we can sidestep this problem by simply specifying the user goals. Because these goals can be specified precisely and unambiguously, the problems typically experienced in scenario based experiments with real users are also avoided.
To summarise, the type of user model developed in section 4 is a goal directed, probabilistic, behavioural model at the intention level.

2.2.3 Error modelling

The problem of modelling the effect of the input system is made surmountable by grouping the different error types together and considering only the total distortion that takes place between the original intention[3] in the mind of the user and the eventual interpretation arrived at by the system. A probabilistic model for these errors can be created by comparing reference transcriptions with recognition output at the intention level.

2.3 Dialogue concepts

This section introduces the structure and basic constituents that will be used to analyse dialogues and user utterances.

2.3.1 Dialogue structure

The dialogue systems under consideration are structured in the sense that they can be described as finite state automata. For this reason, they produce dialogues that have discernible structure, as described by the following concepts.

Utterances: A dialogue consists of an alternating series of user and system *utterances*. For the purposes of this report, the term utterance will be used to refer to the intention level description of what either dialogue participant has said during a dialogue turn. System utterances will also be referred to as *prompts*.

Transactions: At a somewhat higher level, we can see a task oriented dialogue as a sequence of *transactions* between the user and the system. We shall also refer to these as *user goals* that are accomplished. A transaction is completed *correctly* if the system performs the action (such as providing a desired item of information, or performing a desired transfer between accounts) that the user intended it to perform.

[3] The existence of an unambiguous user intention is of course hypothetical, but for the purpose of simulating dialogue it is a useful simplifying assumption. We assume not only that an unambiguous intention exists, but also that it can be expressed in the semantic language of the dialogue system. Where this latter assumption is violated, we classify it as an error by the user in attempting to communicate something that the system cannot understand. The distinction here is between what the user *should have* attempted to communicate (given his/her overall goals and the capability of the system) and what the user did in fact attempt to communicate.

A transaction is carried out in a sequence of dialogue turns (utterances), the possible structures of which are specified by the system design. If a transaction is completed in a relatively small number of dialogue turns, it is said to have been completed *efficiently*.

The goal of dialogue design is to create a system that enables transactions to be completed both correctly and efficiently.

States: We define a *dialogue state* to be a particular set of dialogue circumstances, as characterised by the history of the dialogue (from the system's point of view[4]) and possible external information states such as database feedback.

The dialogue strategy links each state to an action, which is usually a prompt to the user. The prompt is accompanied by a syntax specifying which user utterances the system will be able to understand in the next dialogue turn. More than one state can be linked to the same syntax. For instance, there may be a system parameter specifying whether the system has detected a need for a more detailed system prompt on the user's current options. Different settings of such a parameter amounts to different system states if the term is used strictly. However, it is often useful to use the word more loosely to indicate a group of similar states (ignoring parameters such as the abovementioned one). The syntax is then a useful indicator of the state the system is in.

2.3.2 Utterance structure

An early step in this research was to implement an interface for manual creation of dialogues at the intention level, using only the dialogue system. Thus the speech interface was replaced by a simple point and click interface allowing the user to input intention level utterances directly without the use of speech or words. Some aspects of such utterances are discussed below.

Intentions: A user utterance can be viewed as a sequence of intentions. For simple utterances this sequence will be short, but in systems supporting mixed initiative input an utterance can in principle contain any number of intentions.

Recall the definition of an intention as the minimum piece of information that can be conveyed independently within a given application. An example of an intention in the application discussed here might be "from the savings account", which can occur as a complete utterance in answer to a system prompt for transfer transaction details.

In contrast, the meaning representation in the prototype system makes use of semantic tags. These are similar to intentions, but cannot always occur individually. For instance, the semantic concept "savings account" is defined as a tag, but does not qualify as a valid intention as it cannot occur outside of a context such as "From the savings account", "to the savings account" or "balance of the savings account". However, it is possible to

[4]When recognition errors take place, this will differ from the user's point of view.

define a one to one mapping between intention sequences and valid tag sequences.

Although intentions are autonomous units, they can only occur in specific structures. Like the possible structures of the utterance sequence in a dialogue, the possible structures of the intention sequence in an utterance are specified by the system design. Thus the intention "from the savings account" may be followed by the intention "to the checking account" but not by the intention "balance of the savings account". If a user does produce an invalid intention sequence, part of the utterance may be replaced by the null intention during interpretation so that the input to the dialogue manager is valid.

Choice points: During the construction of an utterance, the user may be faced with a series of choices. For example, the first might be a choice of whether or not to specify a transaction type. The decision to do so then leads to a second choice, namely which type of transaction to specify. This is followed by a choice of whether to specify any further details, and so on. We shall refer to each point where such a choice is to be made as a *choice point*.

Figure 4 illustrates a choice point with three options, plus the null option. Each option results in the addition of an intention (represented by a lettered circle) to the utterance being constructed. If the null option is chosen, no intention is added. The estimation of parameters for a probabilistic user model will amount to estimating the probabilities with which, when faced with a choice point, a user chooses the different available options.

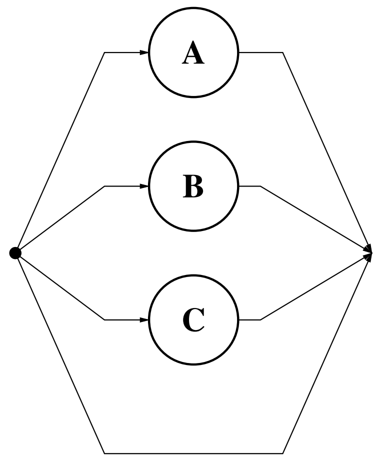

Figure 4: A single choice point.

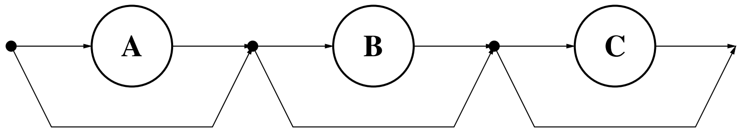

Figure 5: Concatenated choice points.

2.3.3 Examples

Choice points can be combined in a finite state network to model the construction of entire utterances. Figure 5 illustrates an utterance constructed by concatenation of consecutive choice points. Each choice point is entered irrespective of the choice taken at the preceding point. This structure occurs in a situation where the user has the option to communicate any combination from a set of three intentions. The first choice is whether or not to include intention A in the utterance. After making the choice, the user has to repeat it for intention B and then C. This kind of utterance is common in form filling parts of mixed initiative dialogues, which provide the option of specifying extra details that have not been prompted for.

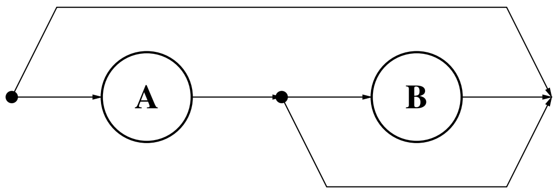

Figure 6: Nested choice points.

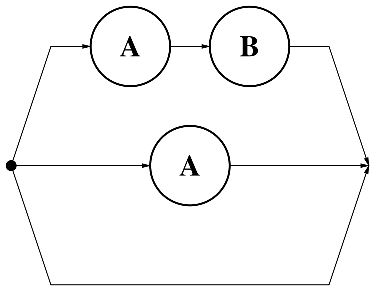

Figure 7: Equivalent structure for nested choice points.

Figure 6 presents a more complicated structure, where the choice made at one choice point influences which choice points are visited further on in the network. An example of this structure in the banking application is the stock quote enquiry, which allows the user to ask for the prices of zero, one or two stocks. This could also be represented as three options in a single choice point, shown in Figure 7. The second representation implies that the choice is made in a single stage, while the nested structure of the first implies that the choice of asking for a second stock is made only after choosing to ask for the first. Representing utterance construction in this way is convenient because, by linking each intention to its own choice point, it separates the individual intentions from the larger context: during parameter estimation it is possible to back off from context to make the estimated probabilities more robust. Also, error generation can be done separately for each intention. These ideas will be explained in more detail in section 4.5 and 4.3.1 respectively.

3 Bigram user model

In this section a simple approach to the user modelling problem is presented. It is argued that a more sophisticated approach is required if realistic simulations are to be produced.

3.1 Model description

A complete user model should contain a mapping from the full set of states the user can be confronted by to the probabilities of the set of allowable user actions.

Eckert, Levin and Pieraccini [1, 2] propose a user model where the state (from the user's point of view) can be defined as the complete history of both user and system utterances. They then simplify by considering only the single preceding system response, creating a bigram model that specifies the probability of each possible user utterance conditioned on the preceding system utterance. Context other than the immediately preceding response is ignored because of data sparsity. In fact, even in the bigram case it was found that the available data in their application was insufficient so that it was necessary to hand-craft some of the parameters.

The approach taken in this section is similar to the one described above. However, rather than estimate probabilities for an utterance as a whole, we view each utterance as constructed by means of a network structure as explained in section 2.3.3. This allows us to estimate separate probabilities for each choice point.

The set of allowable user actions (or *action set*) is defined by the dialogue system. More precisely, it is the set of intention sequences that can be understood by the system. While for many dialogue systems this is constant throughout the dialogue, the prototype system used here links each utterance to a syntax that will be used to parse the user's response.[5] Thus this syntax prescribes a possibly different action set for each user utterance. The action set is described in the user model by a network of choice points, which is constructed from the syntax.

[5]During normal operation, the syntax is used to convert a word level response to an intention sequence. In the simulation, the user model utterances are already at the intention level, so that in this case the syntax acts as a constraint on the action set.

In order to specify the network, it is necessary to specify the list of options at each choice point, with the following information being required for each option:

- The probability with which it is to be chosen. These probabilities can be estimated from data. If a choice point can be used in more than one dialogue state, the probabilities will be state dependent.
- The intention that is associated with this option.
- A list of other choice points that are to be visited as a result of this option being chosen.

Figure 8 illustrates the basic algorithm for generating an utterance using this model. We start with an empty utterance to which intentions can be appended. We also use a stack, called the agenda, on which to store the names of choice points to be visited. The choice points have been initialised to correspond to the syntaxes used in the system. When the agenda has been emptied, the utterance is complete.

3.2 Results and discussion

The weakness of the model described above is that neither the history of the dialogue (other than the most recent utterance) nor the aims of the user are taken into account. Thus, phenomena such as inconsistencies between user utterances and aimlessness on the part of the user are to be expected. These expectations were confirmed by trial simulations using this model. The choice point probabilities for this experiment were hand-crafted rather than estimated from real data. Figure 9 presents an extract from a typical result. (The intention level utterances have been extended to the word level for readability. This includes specifying a particular value for the cash amount rather than a generic term.) This type of dialogue tends to continue for a very long time, with the user continually changing the transaction specification.

It is clear from the example that the user's actions are unnatural: successive user utterances are inconsistent with one another, impossible goals get specified, and the user's behaviour is generally planless. Another common phenomenon was unnecessary repetition (or respecification with different values) of transaction details that had already been specified.

An advantage of the bigram user model is that there is no need to model errors separately. By estimating the choice point probabilities from the output of the recognition/understanding subsystem, a model is obtained that includes both the user and the error models. Also, some applications may only require dialogues with a simple structure, where each transaction is completed within a few turns and there is no relation between separate user utterances. For such dialogues, this user model would be sufficient.

However, many dialogues require specification of more complicated instructions by the user, where different aspects of the specification are interdependent. This is especially evident in applications where the cost of a system misunderstanding is high, so that confirmation subdialogues are needed. The bigram user model is clearly insufficient for such dialogues. The next section addresses this problem by developing a more sophisticated user model, with the idea of imposing an overall structure on the user's behaviour.

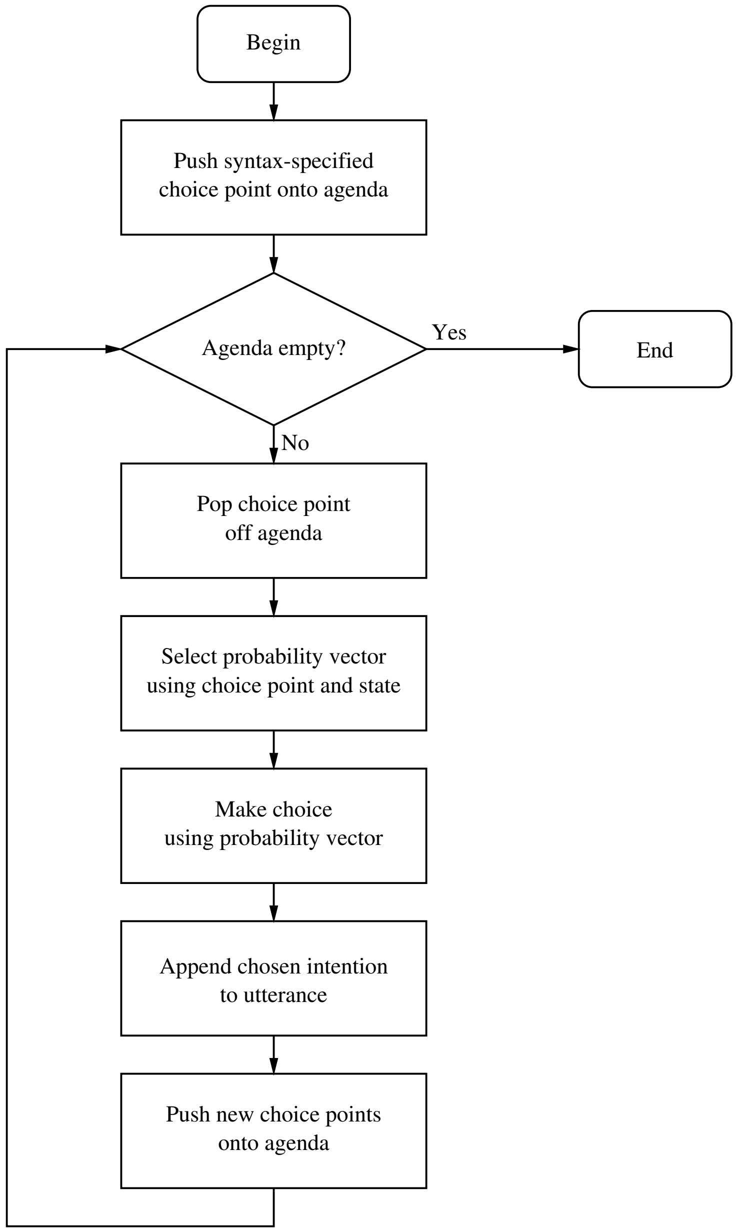

Figure 8: Basic algorithm for generating a user utterance.

System:	Which service would you like?
User:	I want to transfer money to my savings account.
System:	How much do you want to transfer?
User:	500 pounds, from my checking account.
System:	Please confirm the following transaction: 500 pounds to be transferred from the checking to the savings account. Is that correct?
User:	I want to transfer from my savings account.
System:	Do you want to transfer from your savings account?
User:	Yes.
System:	Do you want to transfer to your savings account?
User:	Yes.
System:	It is impossible to transfer from your savings to your savings account. Do you want to transfer from your savings account?
Etc.	...

Figure 9: Extract from a typical simulation produced by the bigram user model.

4 Goal directed user model

This section presents a more detailed user model that was developed to solve the problems encountered by the bigram user model. The user's state is extended by introducing the concept of a user goal. User actions are no longer completely random, but are directed towards achieving the goal. This, along with consideration of specific aspects of the dialogue history, introduces a deterministic component to the user behaviour.

The model is built on the central assumption that the user always attempts to act in accordance with some goal, which remains fixed until it has been completed. Any deviations from such behaviour will be viewed as errors in formulation and are to be modelled along with recognition and understanding errors. These errors are then also incorporated in the simulation.

4.1 Comparison to the bigram model

The model introduced here will be referred to as the goal directed user model (GDUM). The most important difference between the GDUM and the purely probabilistic bigram model is the introduction of the *user goal* concept. User actions are *goal directed*, meaning that they are constrained to be consistent with the user goal. This will have the effect that errors need to be modelled separately: no "incorrect" intentions will be generated by the user model.

Another feature of the model is that it takes previous user and system actions into account. As a result of data sparsity problems, such contextual information could not be built into the bigram user model. However, by deterministically specifying the effect of selected dialogue aspects, it is possible to model the effect of more than just the most recent system prompt.

The GDUM differs from the bigram model in the following ways:

- The user model always has a valid goal.
- Only intentions that are not in conflict with the goal may be appended to the utterance.
- The choices made at choice points have no goal specific content. As a result, the choice point parameters are more general.
- The semantic contents of system questions and confirmation prompts are used.
- Information provided in system prompts regarding goal progress and goal completion is used.
- The system keeps track of which goal details have already been specified so that unnecessary repetition can be avoided.
- The combined effect of formulation, recognition and understanding errors is modelled explicitly.

4.2 Simplified model description

We start by describing a simplified version of the goal directed user model, omitting error modelling, as presented in Figure 10. The terminology used in this figure will be introduced in what follows. Next, section 4.3 will describe an extension of this algorithm to incorporate error modelling.

4.2.1 Transaction tasks and user goals

The different transactions that are supported by the application are defined by means of the transaction type and variables for type-specific details. By filling in the relevant variables, a transaction task can be specified. An example task in the banking application might be to transfer 500 pounds from the savings account to the checking account.
Before each dialogue, the user model is supplied with a list of transaction tasks to be completed. Such a list can be chosen according to the relative frequencies with which users are expected to use the various features of the application, or can be initialised manually if the aim is to simulate dialogues of a specific type. A transaction task can be used to initialise a user goal structure, such as the example in Figure 11, which can then be used during construction of utterances. It is possible, when the application allows more than one task to be pursued simultaneously, for a goal structure to contain details of more than one task. The banking application supports requests for more than one stock quote in the same utterance, so that consecutive stock quote enquiry tasks can be represented in a single user goal. It is also possible for a goal to be achieved partially, in which case the goal can be adjusted to leave only the uncompleted parts of the goal.
During dialogue simulation, the list of transaction tasks to be accomplished is used to initialise consecutive user goals: whenever a goal is completed, it is replaced by the next task (or set of tasks) on the list.

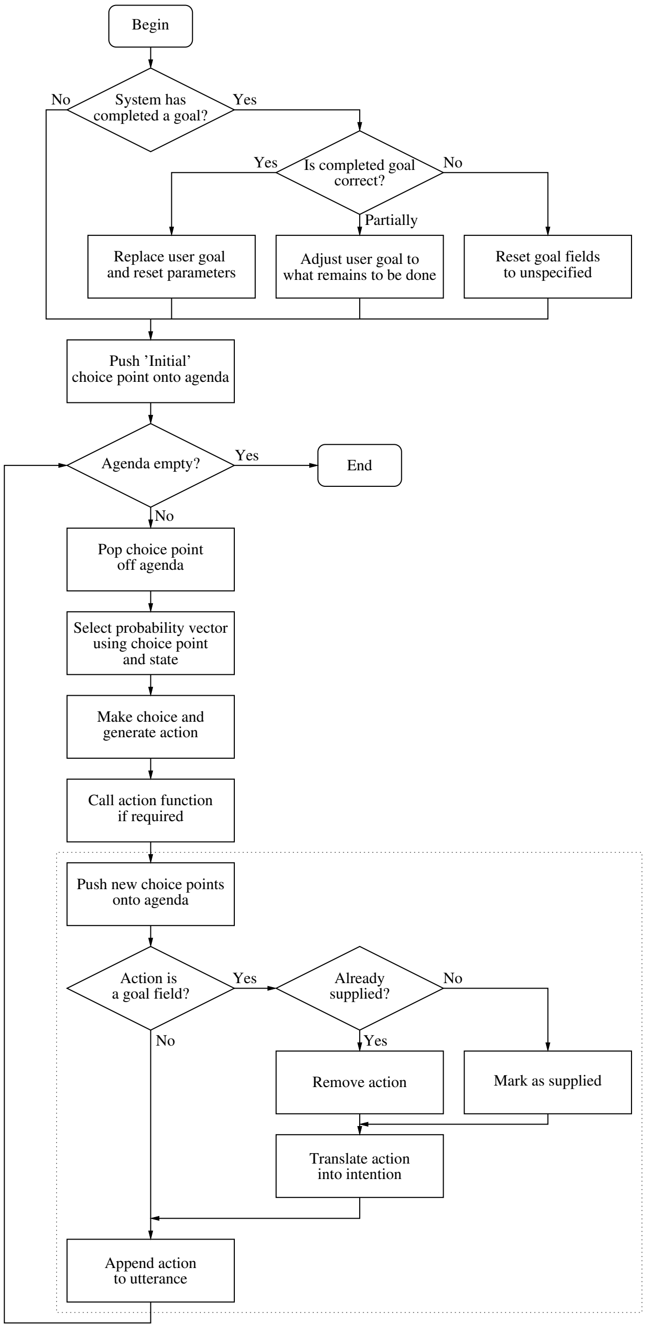

Figure 10: GDUM algorithm for generating a user utterance.

Transaction type:	Transfer
Amount:	500
From account:	Savings
To account:	Checking
Balance account:	NA
Stock name list:	NA

Figure 11: Example of a user goal

4.2.2 Behavioural categories

Not all utterances in a dialogue are directed towards accomplishing the goal. We use the following rough classification of different types of behaviour of a dialogue participant:

Normal: The normal behaviour is to progress directly towards achieving the dialogue goal.

Confused: Either dialogue participant can make mistakes. For example, the system may misrecognise the transaction type requested by the user and prompt for details of the wrong transaction, or the user may not know the system capabilities and say something that the system cannot understand. We shall refer to this as *confused* behaviour.

Error recovery: When a dialogue participant realises that an error has occurred, repair action is taken. For example, the system can reprompt for transaction details in a confirmation subdialogue when the original details are found to be incorrect, or the user can cancel an incorrect transaction that has been initiated.

The first decision in generating a user utterance is which of these behavioural categories the utterance should belong to. Thus the first choice point (named "Initial") makes a choice between the actions "Confused" and "MoveToGoal". The former causes a choice point named "Confused" to be placed on the agenda, so that a further choice can be made as to which of the possible "confused" intentions should be generated. ("Confused" intentions supported in the banking dialogue are "silence", "mumble", "help" and "repeat". The former two are utterances from which the system fails to extract any meaning, while the latter two are commands that can be uttered if the user requires clarification.)
The "MoveToGoal" action invokes a deterministic function that decides first of all whether error recovery is necessary. In the banking dialogue all error recovery is done by uttering the "Cancel" intention. If no errors have been detected, the normal mode of behaviour is adopted, in which progress towards the goal is attempted. What exactly is done depends on the dialogue state and user goal, and typically involves both appending intentions to the utterance (when specifying a given intention is the only permitted way to move towards the goal) and adding choice points to the choice point agenda (for specification of further intentions that are optional but not compulsory).
Clearly, the implementation of the "MoveToGoal" function is highly application specific, as it mirrors the application specific utterance syntax structure. The

idea is that the rest of the model should be completely application independent, so that only this function and the specification of the choice points need to be changed when the model is ported to a new domain.

4.2.3 Using the goal

Whereas the choice points of the bigram user model generated specific intentions such as "transfer" or "balance enquiry", the GDUM first generates more general concepts such as "specify the goal type". These can be produced either by the choice points, or by the deterministic "MoveToGoal" function.
When the general concept has been decided on, the current goal is used to translate it into specific intentions that correspond to the fields in the goal.

4.2.4 Role of the choice points

Most of the choice points in the GDUM have a very different role from that of choice points in the bigram UM. For systems that support no mixed initiative, the "Initial" and "Confused" choice points mentioned above would be enough, as all decisions concerning which transaction details to specify would be determined by the syntax and the goal. However, a mixed initiative system requires further choices to be made by the user, namely which of the possible details to specify and which to withhold. This is what most of the choice points in the GDUM are used for.
For example, the "MoveToGoal" function may decide to specify the type of a transaction as "balance enquiry", and then add the "Balance" choice point to the agenda. The purpose of this choice point is to decide whether or not the name of the account of which the balance is sought should also be added to the utterance.
Another choice point that is required is encountered when, after a transaction has been completed, the user is asked if a further transaction is desired. A mixed initiative choice point named "Carry-on" was introduced to describe the probability with which the user will specify details of the next transaction as part of a positive reply. If this is done, the MoveToGoal function has to be called a second time to determine which details will be specified.

4.2.5 Using information from the dialogue history

A dialogue system can be expected to give the user some feedback on the progress of the dialogue, which can be incorporated into the user model. The following types of feedback are typical:

- **Indirect confirmation:** This takes the form of running feedback on what the system is trying to do, ie. the system does not directly ask the user whether it has understood correctly, but expects to be interrupted if an error has been made. In the banking application, the type of transaction is communicated in this way, and the user has the option of backtracking by way of a "cancel" command.
- **Direct confirmation:** When the cost of an error is high, the system directly asks the user whether it has understood correctly. In the banking application, transfer transactions are confirmed in this way before they are finalised.

- **Feedback on completed goals:** Goal completion feedback happens automatically for information dispensing transactions by means of the system outputting the desired information. For other transactions, the system will specify explicitly when the transaction has been completed.

The GDUM extracts each of these types of information from the system communications and adjusts its actions accordingly. The first case is handled by entering the error recovery mode, the second by comparing the specified details to the goal to determine the correct answer to the confirmation question, and the third by adjusting the current goal according to whether or not it has been completed correctly.
The GDUM also takes the history of its own utterances into account, to avoid unnecessary (and unrealistic) repetition of previously specified goal details. When a goal field is specified for the first time, it is marked as having been supplied so that it will not be repeated later in the transaction. If a system misunderstanding takes place, the relevant fields are again marked as needing to be supplied.

4.3 Error modelling

One of the central problems in speech based dialogue is the high rate of recognition and understanding errors. It is important, therefore, to include a model of such errors in a simulation of the dialogue.
The error model is based on estimating the probabilities of substitution errors by comparison of parser output from hand transcribed and recognised versions of real data. These parameters are then used to generate realistic substitutions of the intentions produced by the GDUM. Deletion errors are treated as a special case of substitutions, while insertion errors are ignored in the current implementation.

4.3.1 Basic description

The process of generating errors is complicated by the fact that the final utterance has to be syntactically valid. Thus, the intention sequence "balance-enquiry, savings-account-balance" cannot be replaced by either "balance-enquiry, from-the-savings-account", or "transfer-transaction, savings-account-balance". In both cases, the context inside the utterance prevents a particular confusion of intentions.
However, modelling errors in the full utterance context is impractical because of the amount of data that would be required. For this reason, we model the errors made on isolated intentions, taking the dialogue context (state) into account, but neglecting the effect of neighbouring intentions in the utterance. It is because errors are modelled on individual intentions that the utterance structure is defined such that a choice point can never produce more than one intention. The construction is kept syntactically valid by simply constraining intentions later in the utterance to be consistent with earlier ones. Thus, once "balance-enquiry" has been conveyed correctly, it serves as a constraint on which errors may occur later in the utterance: "from-the-savings-account" is an error that would be disallowed in this case. In the second example, when "balance-enquiry" is misrecognised, the rest of the utterance is forced to be consistent with that

mistake by discarding the inadmissible "savings-account-balance" and generating a valid utterance conclusion instead.

The approach used here is to perform error generation concurrently with the generation of the utterance. That is, when an intention has been decided on by the GDUM, but before it is added to the utterance, it is passed to the error generation module. This module may then perform a substitution (or deletion) on the intention to produce the final version. In this way, production of utterance segments that may later become invalid is avoided.

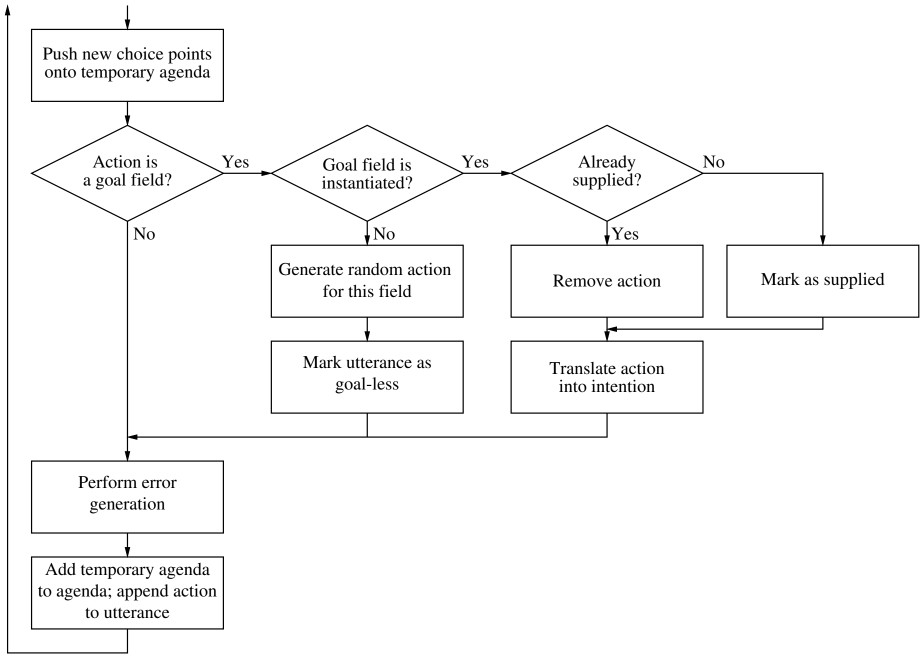

Figure 12: Adaptation to the GDUM algorithm to include error generation.

Generating errors in the same step as the original intentions necessitates some changes to the GDUM algorithm. These changes are shown in Figure 12, which replaces the section inside the dotted line of the GDUM flowchart (Figure 10). The most important change is the addition of the error generation block, of which the details are shown in Figure 13. We shall discuss this part of the algorithm first before returning to its integration into the rest of the system.

4.3.2 Substitutable intentions and substitution choice points

The error generation algorithm is applied to all substitutable intentions generated by the GDUM. Intentions are regarded as substitutable if they are not non-speech acts such as hanging up, silence, or the null intention, and the utterance has not been marked as "goalless" (explained later). Because substitutions are not performed on the null intention, insertions are neglected. Since insertions are very rare in the data, estimating insertion probabilities for each possible

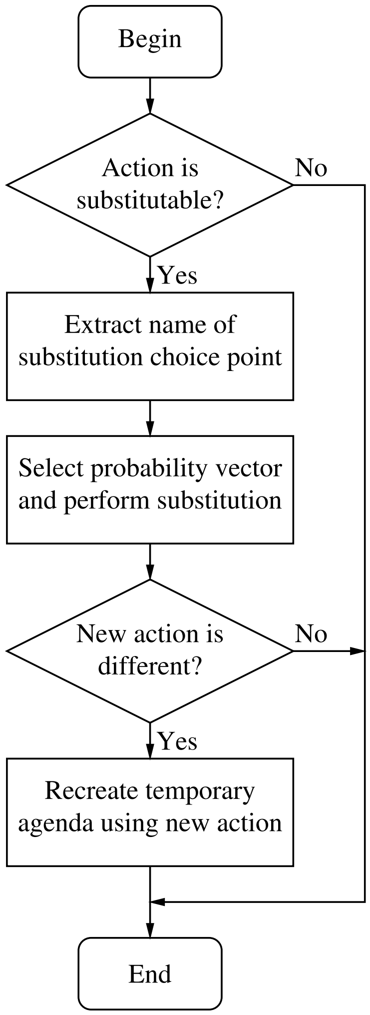

Figure 13: Error generation algorithm.

intention would be impractical. Also, it is unlikely that insertions would occur often enough to have a noticeable effect on the performance of the model.
A choice point is defined for each substitutable intention in each of the syntaxes used by the system. These choice points, referred to as *substitution choice points*, describe the probabilities with which the specified intention can be substituted for any of the allowable substitute intentions, including itself (no substitution) and the null intention (a deletion). Separate choice points are defined for the different syntaxes, since these determine which substitution options are available. Because syntax is not the only factor that identifies a dialogue state, the choice points may also contain different probability vectors for different dialogue states, where applicable.

4.3.3 Intention groups

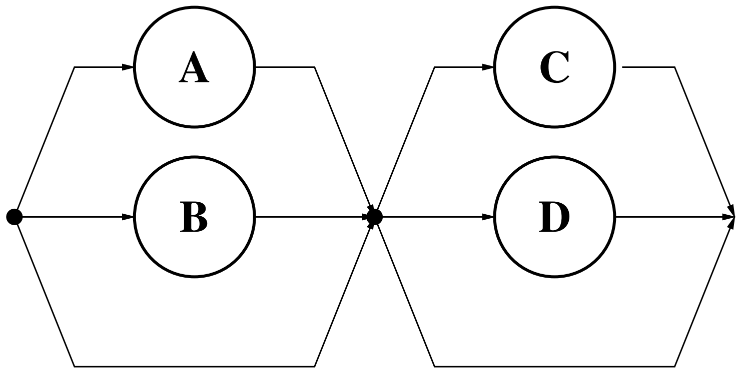

Figure 14: Example utterance structure illustrating the alignment problem.

In order to estimate substitution probabilities it is necessary to align the reference intention sequence with the recognised sequence so that corresponding intentions in the two sequences are matched. Consider the partial utterance structure shown in Figure 14. This is an utterance that may contain (optionally) either of intentions A and B, followed by (optionally) C or D. Suppose the reference sequence is AC, and the recognised sequence is B. The question is how to decide whether the intention B should be aligned with A or with C. Both alignments are possible: the recogniser might have made a deletion error on either of the original intentions and a substitution error on the other. This type of problem is well known in the theory of sequence comparison and can often be solved by developing some dynamic programming algorithm. In this case, such an approach would pose further problems such as the definition and estimation of a distance metric for intentions, which does not seem practical.
The approach taken here is to consider only substitutions within restricted groups of intentions. These groups, called *intention groups*, are chosen to correspond with the intentions allowed at specific choice points in the syntax, using the same choice point specification as for the bigram user model. In the example, it would mean that B could only be a substitute for A and not for C, as it is not an option to be produced at the same choice point as C. One intention

that is always present in an intention group is the null intention, which causes nothing to be added to the utterance. In the example, the interpretation would be that C has been substituted for the null utterance, which is the same as saying that a deletion error has taken place at that point.
This approach fails to make provision for the possible example where the reference utterance is A and the recognised utterance is C. This would be regarded as an insertion-deletion pair rather than a substitution, which is not the ideal interpretation, but still allows the example to be modelled.
By using the above approach, the options available at the choice points are limited and the estimation of substitution probabilities made possible. During error generation, the substitution choice points can then be used to perform probabilistic substitutions on every intention produced by the GDUM. Most of these substitutions will have no effect, as an intention will usually be more likely to be substituted by itself than by any other intention. If the new intention is different from the original, however, a recognition/understanding error is simulated by appending the new intention to the utterance rather than the "correct" one. Since the rest of the utterance must be consistent with the new intention, the items to be added to the agenda are also changed accordingly.

4.3.4 Wandering off the goal

When substitutions take place in the course of utterance generation, it is possible to reach a situation where no relevant goal fields are instantiated. This happens, for instance, if a substitution error has been made on the transaction type. The goal may be a balance enquiry, with the specified transaction type being "transfer", which may lead to the GDUM calling for the specification of a transaction detail such as the "from" account, which is not specified in the goal.
Such a situation is handled by supplying a random detail of the required type. As there is no point in performing substitutions on intentions that are already random, the utterance is then marked as "goalless" so that no further substitutions need to be made.
The specification of random details is done by means of yet another type of choice point, called *no goal* choice points. These have the same structure as the choice points used for the bigram user model, listing all possible choices for a particular point in the syntax.

4.3.5 Tag and word level information: hiding the details

It is often not practical to include all the information of an utterance in the intention level description. A typical example is the specification of a cash amount: rather than define different intentions for each possible amount, the intention language contains only a single intention to indicate that the amount was in fact communicated. The actual details (the numerical value of the amount) are communicated at the word level only. The semantic tags used for intentions such as this that have further unspecified content are referred to as *content tags*.
For the purpose of simulation, it is again desirable to hide this type of detail. The need for generation of word level utterances is eliminated by defining an additional intention for each content tag, to allow for the case where the details

are misrecognised. Thus, both the intentions "cash amount" and "bad amount" are defined. Only the former intention can be generated by the GDUM, but during error generation it is possible for it to be replaced by the latter. When one of these intentions is placed in an utterance, an appropriate amount is automatically specified at the word level. For "cash amount", the amount in the current goal is specified, and for "bad amount" a default incorrect amount is used. Other details handled in this way for the banking application are pin numbers and stock names: in both cases, it is not practical to estimate substitution probabilities for each possible pair of content details.

4.4 Example

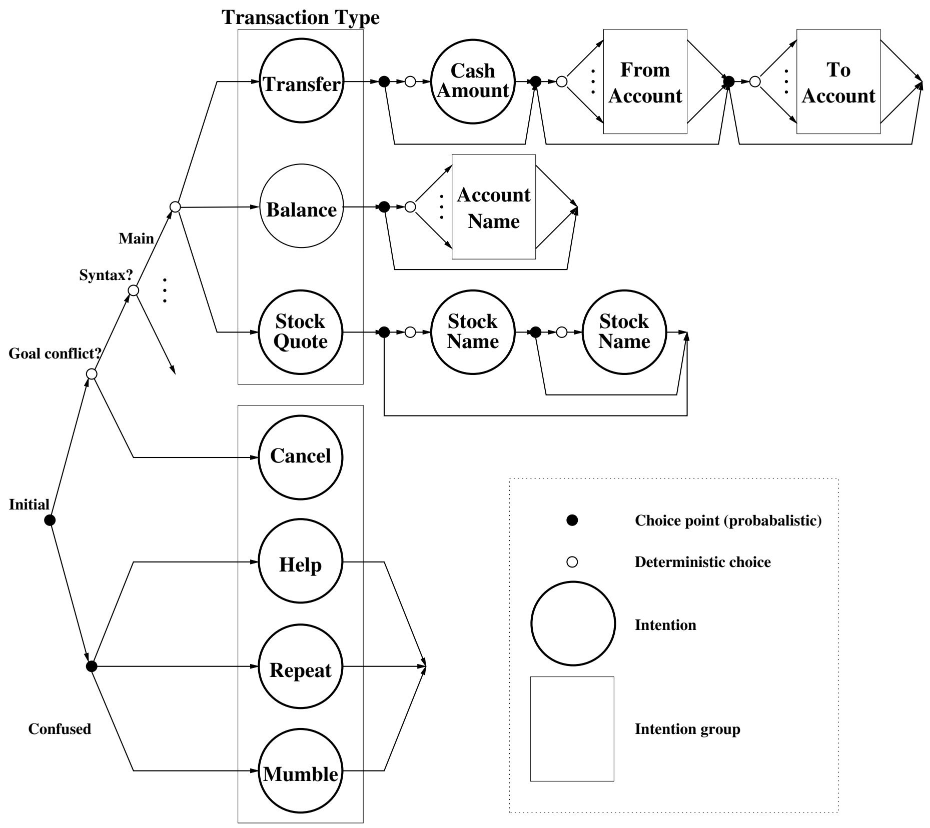

Figure 15: Partial structure for utterance construction in the banking application.

To demonstrate how the concepts introduced above are used in an actual application, the utterance construction for the banking application is illustrated in figure 15. The figure shows the "initial" choice point, which leads to either a series of deterministic decisions or the "confused" choice point. The first deterministic decision is based on whether there is a conflict between the user goal and the inferred system goal, which would cause the utterance to consist of sim-

ply the “cancel” intention. Note that the “cancel” intention belongs to the same intention group as the “confused” intentions. This encodes the fact that these intentions can always be substituted for each other during error generation.
If there is no conflict, the next decision depends on the allowable syntax, with each syntax determining the structure of what follows. Shown in the figure are the possible choice sequences for the “Main” syntax, which is used when the user has not yet specified a type for the current transaction. The deterministic choices from this point onwards are specifications of a particular goal field. The options for each such choice belong to the same intention group. In the case of the content tags “Cash Amount” and “Stock Name”, the deterministic choice determines the specific tag contents, but not the intention tag itself.

4.5 Parameter estimation

While it is possible to initialise a user model with hand-crafted parameters, if task specific data is available it is desirable to make use of it to estimate the probabilities with which users choose different options at the choice points, and with which errors occur.

4.5.1 Description of parameters

The choice points required in the banking application have already been discussed. The type of parameters we need to estimate for them are:

“Initial” choice point: The probability of the user behaviour for a particular utterance being normal or confused. In the current version of the model, this is dependent on the most recent system utterance, but not on previous user utterances.

“Confused” choice point: The relative probabilities of the different types of confused utterances. Again, this is treated as being independent of previous user utterances.

Mixed initiative choice points: The remaining “ordinary” choice points model the user’s behaviour in mixed initiative situations. They contain the probabilities with which a user specifies optional transaction details. These probabilities are modelled as dependent on both the type of detail being specified and on the system prompt, but not on the other decisions taken in the same utterance.

Substitution choice points: The parameters to be estimated for substitution choice points are confusion probabilities of different intentions. Within each intention group, the probability with which the recognition/understanding process will replace any given intention with any other must be estimated.

Note that all probabilities are, in principle, dependent on the dialogue state, as defined by the most recent system prompt. However, if data is sparse, robustness can be preserved by estimating more general, state independent parameters. It would also be possible to generalise over the specific type of transaction details for mixed initiative choice points, although that has not been implemented in this research.

4.5.2 Data requirements

The validity of estimating model parameters from real data rests on the assumption that the average user knowledge about the mixed initiative functionality in the system is the same in the data collecting experiment as in the dialogues we want to simulate. This is somewhat problematic, since system development corpora will usually include dialogues conducted by system developers who know more about the system functionality than average users and who even go out of their way to test aspects of system functionality that they would not be used often under normal circumstances. In spite of this, we proceed in the hope that the available data will prove adequate in most cases.

The data to be used is an intention level transcription of dialogues between real users and systems in the task domain for which the user model is being constructed. Ideally, the transcriptions should be obtained by parsing hand transcriptions of the recorded speech. If that is not available, the results obtained from transcriptions produced by the speech recogniser may also be adequate. For each user utterance, the allowable syntax and the preceding system prompt have to be known.

While, for the experiments reported here, the system used to collect the data was the same as the one to which the user model was interfaced during simulation, this need not be the case for the model to be valid. However, differing strategies would cause some problems with parameter estimation. For example, one aspect of the system strategy is that, during transfer transactions, the "to" account was never prompted for before the "from" account had been input. Thus statistics on how often users would supply both account names when prompted for the "to" account are not available. Similarly, the system does not allow any details to be specified in response to a direct question unless those details include the information that has been asked for. The data does therefore not contain the information needed to estimate how often this would happen if a strategy were used that does allow this.[6]

If domain specific data is unavailable, it may be possible to use parameters estimated on data from other domains. It is a property of the GDUM that the parameters are not inherently domain specific. However, it is still to be expected that the parameters measured in different domains will be different. It remains to be seen how large these differences will be.

4.5.3 Estimation of choice point probabilities

The first step in estimating choice point probabilities is to produce counts of the number of times the choices in question are taken. This is done for each system prompt by taking into account the associated syntax. A user utterance is analysed by going through the steps that would be taken in constructing it. Each time a choice point would have been put on the agenda, the total occurrence count of that choice point is incremented. By investigating the utterance it can be seen which option was in fact taken, so that the count of that particular option can also be incremented. This analysis is implemented using simple string matching techniques.

[6]If a user attempted such an utterance, it would be unparseable and would be classified by the system (and transcribed at the intention level) as a "mumble" utterance.

A specific problem that occurs here is that of distinguishing between positive and negative answers to yes/no questions. This is nontrivial, because such questions are often not answered explicitly. Rather, a positive or negative answer is implied by the rest of the utterance, which has to be interpreted in context. This is a classic problem that has received attention in the linguistic study of human-human dialogues [6].

Fortunately, the yes/no problem is linked specifically to the word level. When the work is done at the intention level, we need not distinguish between explicit and implicit positives or negatives, so that the problem can largely be side-stepped. While analysing the data, knowledge of the dialogue system design was used to decide, for each state, whether the absence of a yes/no keyword should be interpreted as an implied positive or an implied negative. Thus, specification of details in the absence of yes or no was treated as an implied negative in confirmation subdialogues, and as an implied positive in the carry-on choice point. These decisions mirror the way such utterances are interpreted by the system.

Once the counts have been obtained, maximum likelihood estimates of the choice point probabilities are calculated:

$$P(O_{ij}|CP_i) = \frac{cnt(O_{ij})}{cnt(CP_i)}, \tag{1}$$

where CP_i is the ith choice point, O_{ij} is the jth option of the ith choice point, and $cnt(\cdot)$ is the counted number of occurrences.

Data sparsity was not encountered during estimation of these parameters in the banking application. If sparsity does occur, it is possible to smooth the parameters by counting choice point occurrences in a state independent fashion. The resulting model would have parameters that are less specific, but more robust against data sparsity.

4.5.4 Estimation of substitution probabilities

As for ordinary choice points, probabilities for the substitution choice points are estimated by obtaining counts and applying maximum likelihood estimation. The counts are obtained by comparing reference and recognised versions of the intention level data as follows:

When a given intention is found in the reference, it is searched for in the recognised version. If found, it is assumed to have been recognised correctly. (This assumption is valid if it is true that the same intention cannot appear twice in any one utterance.) If it is not found, a valid substitute is searched for. The first such substitute is assumed to be the recognised version of the original utterance. (Here the assumption is that no two members of the same intention group will appear in the same utterance.) If no valid substitute is found, the intention is counted as having been substituted by the null intention.

A separate estimation is needed for the details obscured by content tags. Here the required parameter is the probability that, given that a content tag was recognised correctly at the intention level, the word level details were also recognised correctly. This is estimated by using word level data and comparing the correctness of entire sentences. In some cases, these sentences have to be selected carefully in order to ensure that the results are not biased by misrecognition

of words that do not have relevant semantic content.
Some of the parameters estimated in this way were subject to data sparsity. This problem was treated by generalising those choice points for which the number of instances in the data was below an arbitrary threshold. Instead of using state dependent probabilities for these choice points, data from different dialogue states were pooled together to enable estimation of state independent probabilities.

5 Results

5.1 Experimental setup

The purpose of the experiments reported here was to ascertain to what extent the simulated dialogues give an accurate picture of the dialogues produced by real users applying the system in the task domain. The problem in doing this is that it is in fact impossible to obtain such "real" dialogues for a system that is still under development and therefore not yet fit for deployment in the task domain. (Recall that this problem is one of the main motivations for simulating the dialogue in the first place.)
In spite of this, a data corpus was created by a set of callers and used to calculate the average number of dialogue turns needed to complete the various types of transactions possible within the system. The same transactions were also simulated a large number of times, so that the simulated results could be compared with those obtained from real callers.
Unfortunately, it was not possible to ascertain which of the achieved goals in the real dialogues were in fact what the users intended and which were the results of system misunderstandings.[7] Thus goal achievement rates could not be calculated.

5.2 Data corpus

The dialogue system on which these experiments are based, was tested by a group containing both system developers and members of the public. While the latter were given no instructions on the system other than a list of valid pin numbers, most of them were familiar with a touch-tone telebanking system offering similar services.

Goal directed dialogues	316
Completed ID but no further goals	174
Failed to complete ID	351
Total	841

Table 1: Dialogues in the corpus.

[7]While this problem can in principle be addressed by asking callers to act out specific scenarios, such an approach has also been found to present problems[7].

User identification	490
Transfer transactions	243
Balance enquiries	282
Stock quote enquiries	575
Dialogue exit	133

Table 2: Distribution of goal types in the corpus.

Details of the resulting corpus are given in table 1. The corpus consists of 841 dialogues, but this includes various tests that were in fact not proper dialogues. Only in 316 of the dialogues did the users actually complete any transactions. When unparseable utterances such as silence or mumbles are discarded, the remaining corpus contains 4201 utterances.

The number of completed goals of each type is given in table 2. Apart from the main goals representing the services provided by the system (transfer transactions, balance enquiries and stock quote enquiries), two more goals were defined. The user identification (ID) goal is achieved whenever the user enters a valid pin number and is accepted by the system. The dialogue exit goal is achieved when the dialogue terminates normally, after the system outputs its final greeting prompt. This latter goal was often not reached, as users hung up in mid-conversation. However, it does give an indication of how much time was spent by users asking for repeats and/or help after completing a goal but before deciding to terminate the conversation.

The nature of the corpus is heavily influenced by the facts that most of the data was created by a small subset of callers, and that the callers tended to explore rather than use the system. This is evident in phenomena that would not normally be expected in a corpus created by callers actually using a system, such as long sequences of consecutive transactions of the same type in a single dialogue. One user, for instance, navigated through the dialogue for 25 turns without completing any goals. For this reason, although it is useful for analysis at the utterance level, the corpus is not highly suitable for drawing conclusions about high level user behaviour or efficiency of the high level design.

A subset of 219 dialogues was isolated for testing purposes. The test set contains 140 dialogues in which at least one goal was accomplished, and a total of 1037 parseable utterances. The remaining utterances were used as a training set to estimate the system parameters. The training set contains 350 dialogues accomplishing at least one goal, and a total of 3164 parseable utterances.

5.3 Parameter estimation

The estimated parameters included:

- A total of 30 probabilities for 9 mixed initiative choice points, some of them using more than one context.

- A total of 418 probabilities for 85 context sensitive substitution choice points.

A few parameters were estimated separately, namely silence and mumble probabilities, and recognition probabilities for pin numbers and cash amounts (the values of which were not specified in the tagged data).

The initial implementation of the model used choice point probabilities based on guesswork. When probabilities were estimated from real data, some of the parameters were found to deviate substantially from the early guesses. This underlines the importance of having a good data source available.

Some of the parameters were found to be very sensitive to context. For example, during specification of a transfer transaction, the "to" account was specified 17% of the time after prompts for the "from" account, but only 5% of the time after prompts for the cash amount. Such variations across context seemed to be larger than the variation across different types of transaction. However, larger experiments with more reliable data sets would be required before any definite conclusions can be drawn on this issue.

5.4 Evaluation methodology

Commonly used criteria for evaluating dialogue systems are the average number of dialogue turns required to complete a goal successfully, and the failure rate of the dialogue. For this reason, it makes sense to base the evaluation of the simulation on whether goal completion times and failure rates for the simulated dialogues correspond to those measured on real dialogues.

Different types of failure can occur in the dialogue, and are monitored during dialogue simulation. Failures for the banking application were defined as follows:

User hanging up It is possible, when the system asks the user whether any more transactions are required, for a negative response to be misrecognised. In such a case, the user will be asked for transaction details in spite of the fact that there is no instantiated goal. This case is treated by causing the user model to hang up immediately. This is the only situation in which the GDUM will hang up before formal completion of a dialogue.

Transaction failure Two types of incorrect transaction completions can occur. When the system completes a transaction by retrieving some information that has not been requested, it is simply ignored. The cost of such an error is that the length of the dialogue is increased by the system performing a wrong but otherwise harmless task.

Transactions like inter-account transfers, however, result in the system performing some action that changes the external world. Mistakes in such transactions are highly undesirable, and should be penalised more seriously when system performance is evaluated. For this reason, incorrect transaction completions of this type are defined as *transaction failures.*

Dialogue failure A dialogue system should have a mechanism by which it terminates the dialogue when it detects that no progress is being made. Typically, when it fails to extract any meaning from a number of consecutive user utterances, the system will hang up or connect the user to a human operator. This situation is defined as *dialogue failure.*

5.5 Dialogue simulation

Three user scenarios were created for the simulation experiments, so that the number of turns taken for all the subgoals in the system could be measured. The scenarios were:

1. **Transfer:** Transfer 500 units from the savings account to the checking account.
2. **Balance:** Find out the balance on the savings account.
3. **Stock quote:** Find out the stock prices for 3 specified stocks (thus three goals are achieved in this scenario).

Goal	Real dialogues	Simulated dialogues
User identification	2.01	1.51
Transfer transactions	7.97	6.20
Balance enquiries	3.38	2.20
Stock quote enquiries	3.14	2.09
Dialogue exit	2.17	1.29

Table 3: Goal achievement times (average number of turns) for real and simulated dialogues.

A set of 1000 simulated dialogues was created for each scenario. The resulting average number of turns for the different goals are given in table 3, along with those obtained from the real data. While the numbers are not identical, the ordering of the different goal types correspond (except for dialogue exit, which is quicker than user identification in the simulated dialogues). It is clear from both data sets that transfer transactions take much longer than simple enquiries.

Dialogue failure	0%
Transaction failure (transfer transactions)	4.6%
Hang up rate	10.3%

Table 4: Failure rates for simulated dialogues.

The failure rates (as defined in section 5.4) of the simulated dialogues are given in table 4.

5.6 Conclusions

The following observations can be made:

- Goal achievement in the simulations is consistently faster than for real users, by up to one turn per goal. The main reason for this seems to be that real users are considerably less goal directed and more confused about what they want to do than the user model. One of the basic assumptions of the user model is that the user always knows what he/she wants to do next. This is of course not true for real users, and especially for the unmotivated callers who created the data set. It is not clear to what extent this phenomenon would be observed if the simulations are compared to data gathered during actual use of the system.

- In spite of the enquiry types having similar dialogue structure, stock quote enquiries are slightly faster than balance enquiries. This could be due to the fact that the system allows two stock quote enquiries to be performed in the same utterance. In the simulations, this appears to have outweighed the fact that stock names are confused more often than account names.

- In both real and simulated data, the user identification goal takes rather longer than one would like. This is due to the fact that a high confusion rate of 27.2% was experienced for pin numbers.

- The hang up rate of the simulated user is surprisingly high. Recall that the user model only hangs up when it is prompted for transaction details while it has no goal. This happens after a negative answer to the yes/no question determining whether the user wants to continue the dialogue, is misrecognised as positive. Further investigation revealed that the measured confusion between a negative and an implied positive answer is indeed correspondingly high. This is due to the linguistically complex nature of the implied positive.

- The transfer transaction failure rate of 4.8% is unacceptably high. This is mainly due to the high recognition error rate on cash amounts (18.7%), and the fact that the system strategy causes the wrong amount to be transferred when an amount is misrecognised twice in succession. This is a flaw in the dialogue strategy that escaped the developers' attention until it was highlighted by the simulation.

Inasmuch as the real data gives some indication of what should be expected from users, it is gratifying that the real and simulated data display similar characteristics. However, it should be emphasised that the real data cannot be taken as providing a definitive guide to user behaviour. In fact, it may be argued that the simulations produced by the system described here should be more useful than the actual data for dialogue level analysis.
Some of the observations listed above throw useful light on system aspects that ought to be investigated. Without the simulations, it would be a much more arduous task to isolate these aspects. Thus it is clear that the simulation system would be a useful development tool for spoken dialogue systems.

6 Summary

This report presented the development of a system for simulation of man-machine dialogues. The central components of the system are a goal-directed model of user behaviour and a model of recognition/understanding errors. The simulations produced by the system were found to be realistic, and showed reasonable agreement with real dialogues in terms of average length for individual dialogue tasks.

The simulated dialogues also provided some useful hints as to which points of the dialogue system under investigation needed further attention, indicating that a simulation system would in fact be a useful tool for SDS development.
The immediate applications of the system include optimisation, evaluation and testing of dialogue systems during the development phase. Apart from these applications, it is envisaged that it will be used as a tool for further work on automatic design of dialogue strategies.

References

[1] W. Eckert, E. Levin, and R. Pieraccini, "User modelling for spoken dialogue system evaluation," *Proc. IEEE ASR Workshop*, 1997.

[2] W. Eckert, E. Levin, and R. Pieraccini, "Automatic evaluation of spoken dialogue systems," tech. rep., AT&T Labs Research, 1998.

[3] R. Sutton and A. Barto, eds., *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts; London, England: MIT Press, 1998.

[4] E. Levin, R. Pieraccini, and W. Eckert, "Using markov decision process for learning dialogue strategies," *Proc. ICASSP*, 1998.

[5] M. Araki and S. Doshita, "Automatic evaluation environment for spoken dialogue systems," in *Dialogue Processing in Spoken Language Systems* (E. Mayer, M. Mast, and S. LuperFoy, eds.), pp. 183–194, Springer, 1997.

[6] B. Hockey *et al.*, "Can you predict responses to yes/no questions? Yes, no, and stuff," *Proc. Eurospeech*, pp. 2267–2270, 1997.

[7] J. Sturm, E. den Os, and L. Boves, "Issues in spoken dialogue systems: Experiences with the dutch arise system," *ESCA Workshop on Interactive Dialogue in Multi-Modal Systems*, pp. 1–4, 1999.