REGULAR PAPER SECTION

DEVELOPMENT OF A SPOKEN DIALOGUE SYSTEM OPERATING IN AFRIKAANS, SOUTH AFRICAN ENGLISH AND XHOSA

M. Tait*, A. Visagie* and T.R. Niesler*

** Department of Electrical and Electronic Engineering, University of Stellenbosch, South Africa.*

Abstract: We present a spoken dialogue system operating in Afrikaans, South African English and Xhosa. The architecture of the overall system as well as the speech recognition, natural language understanding, dialogue control and speech synthesis subcomponents are described. Prototype systems operating in each of the three languages have been developed and subjected to user trials. The performance achieved by these systems is compared, giving some insight into the use of each language in spoken dialogue systems.

Keywords: Spoken dialogue system, speech recognition, speech synthesis

1. INTRODUCTION

Speech is the most natural form of human communication. For this reason speech-based computer interfaces have the potential of making the interaction with machines both easier and more intuitive.

A **Spoken Dialogue System** is a machine designed to maintain a verbal dialogue with a human user. The aim of the dialogue is to negotiate a certain task with a user using speech (as opposed to keyboard and screen). This task is often to provide the user with particular information, or to effect a transaction on the user's behalf.

The following figure illustrates the components of our dialogue system and how it operates.

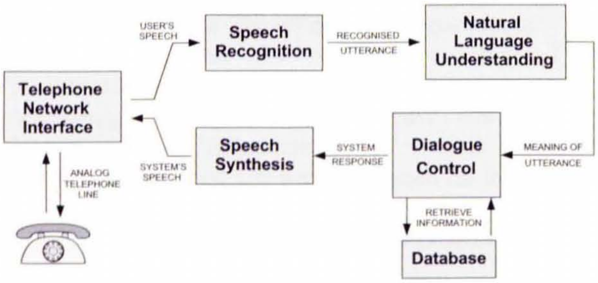

Figure 1: Architecture of a spoken dialogue system.

The user contacts the system by telephone. The speech recognition component transcribes the acoustic signal received from the user into a sequence of words. These words are interpreted by the natural language understanding component to determine the meaning of the user's utterance. Based on this extracted meaning, the dialogue controller decides on the most appropriate next action, which may include querying a database in order to retrieve information requested by the user. Finally, the system responds to the user by means of the speech synthesis component. This cycle continues until the transaction has been completed.

A hotel reservation task has been chosen as an easily-understood and concrete example with which to develop and test our dialogue system technology. The system allows the user to make a (fictitious) hotel reservation over the telephone. The goal is for the system to interact with the user in much the same way a human receptionist would. A typical extract from the dialogue may proceed as follows:

1. The system asks the user to supply some information that it needs in order to proceed with the dialogue. This requires **speech synthesis**.

   Example: "*When will you be arriving at the Mount Elegance Hotel?*"

2. The systems listens for the users reply. This requires **speech recognition**.

   Example: "*Umm, I'll be there on the 6th of October.*"

3. The system interprets the user's reply. This requires **natural language understanding**.

   Example: "ArrivalDate = 06:10:2005"

4. The system decides on the the most appropriate next action to take. This requires **dialogue control**.

   Example: Request departure date.

5. The system synthesises its next utterance based on the next action.

   Example: "*And when will you be leaving?*"

This cycle of system prompting followed by user response continues until either a hotel booking is completed successfully, or the user becomes dissatisfied and terminates the dialogue prematurely.

The work presented in this paper was carried out as part of the recently completed African Speech Technology (AST) project [14]. The AST initiative was aimed at promoting the technological development of the indigenous languages of South Africa. One of its outcomes was the preparation of speech corpora in five of the country's eleven official languages. These corpora were used to develop the acoustic models used by our spoken dialogue systems.

In the following we will describe the particular architecture that we have adopted for the construction of spoken dialogue systems. This is followed by an appraisal of the system's performance when applied to the hotel reservation task in Afrikaans, in English and in Xhosa.

2. DIALOGUE SYSTEM ARCHITECTURE

The functionality discussed in the previous section is implemented as a set of five software components, namely the System Manager, Telephony Manager, Recognition Manager, Dialogue Controller and Prompt Generation Manager. These components interact via inter-process communication (IPC) and work together to form a framework within which a spoken dialogue application can be developed.

2.1. *The system manager*

The system manager initialises and configures the other software components. Our architecture allows several dialogue systems to share the same telephony, recognition and synthesis managers. The system manager coordinates this resource sharing and maintains global system logs.

2.2. *The telephony manager*

The telephony manager is the interface between the dialogue system and the public telephone network, as illustrated schematically in Figure 1. It notifies the system manager of incoming calls, and may be instructed by the dialogue controller to accept such calls. Furthermore the telephony manager relays user speech data from the telephone network to the recognition manager, and system speech from the prompt generation manager back to the telephone network. Finally, once the dialogue is complete the telephony manager can be instructed by the dialogue controller to terminate a call.

2.3. *The recognition manager*

The recognition manager handles both the speech recognition and natural language understanding processes illustrated in Figure 1. These processes are discussed in more detail in sections 3 and 4. User speech data is passed to the recognition manager by the telephony manager. The recognition manager in turn passes the inferred meaning of the utterance to the dialogue controller.

2.4. *The dialogue controller*

The dialogue controller decides what the most appropriate next step in the dialogue should be, based on the meaning obtained from the recognition manager. If a database must be queried, this is done by the dialogue controller. Finally, the dialogue controller formulates a user response in the form of a text string, and passes this to the prompt generation manager. One may regard the dialogue controller as the essence of the spoken dialogue system, and the remaining managers as the controller's communication channels to and from the user. The operation of the dialogue controller is discussed in more detail in section 5.

2.5. *The prompt generation manager*

The prompt generation manager is responsible for the synthesis of a speech waveform from a text string passed to it by the dialogue controller. This speech waveform is in turn passed to the telephony manager for playback to the user. Speech synthesis is discussed in more detail in section 6.

3. SPEECH RECOGNITION

The recognition manager uses the HTK decoder for speech recognition [17]. This open-source hidden Markov model-based (HMM) speech recognizer performs a time-synchronous beam-search using the Token-Passing procedure [16].

A set of HMM acoustic models was trained for each of the three languages using the HTK tools and the AST speech corpora. The speech was parameterised as Mel-frequency cepstral coefficients (MFCCs) and their first and second differentials. Diagonal-covariance speaker-independent cross-word triphone models with three states per model and eight Gaussian mixtures per state were trained using the phonetically-labeled training sets by embedded Baum-Welsh re-estimation and decision-tree state clustering. The performance of these models, in terms of phoneme recognition error, is given in Table 1.

Phoneme set	Number of phonemes	Phoneme recognition accuracy (%)
Afrikaans	83	67.4
English	72	74.7
Xhosa	110	64.3

Table 1: Phoneme recognition accuracies.

A global set of 154 phonemes based on the International Phonetic Alphabet (IPA) has been used to transcribe all AST corpora. The table shows that a different subset of these were present in each corpus, and that the number of phonemes present in each of the three languages varies considerably. This to a certain extent accounts for the differing phoneme recognition accuracies. In particular, the best performance is achieved for English, which has the smallest number of phonemes, while the recognition task is more difficult for Xhosa with its much larger number of phonemes. A more detailed analysis of the phonetic content of each of the three corpora as well as a description of the development of the acoustic models is given in [12]. The English phoneme recognition accuracies reported in Table 1 are similar to those reported by other authors for American English for the Wall Street Journal and TIMIT corpora [10]. A direct comparison is difficult, however, due to the differing natures of the acoustic training and test data, as well as the differing number of phonemes.

Pronunciation dictionaries were created for each of the three languages to allow word-based speech recognition using the phonetic acoustic models. For English, this dictionary was created entirely by human experts. For Afrikaans and Xhosa, initial pronunciations were determined using grapheme-to-phoneme rules, and these were subsequently corrected and validated manually.

4. NATURAL LANGUAGE UNDERSTANDING

Our dialogue system makes use of a finite-state natural language understanding network, as currently successfully employed by a number of experimental as well as commercially deployed dialogue systems [1, 6, 9, 11]. In this approach the set of user responses that can be processed (i.e. "understood") by the system is precisely defined by a graph whose links (edges) correspond to words or sets of words. The set of sentences that the system will accept is given by the set of all the unique paths through the network. As an example, consider the finite-state network illustrated in Figure 2.

This simple network represents a total of eight different sentences which can be identified by following different paths from the start to the end node. In practice the finite-state networks are more complex and the number of utterances they cover much larger.

Certain links in the network have associated *variable assignments*, for example "city=Durban" in Figure 2. The value assigned to each variable after traversing the network from start to end nodes will be passed to the dialogue control unit. Hence the variable assignments effect the process of natural language understanding and the values of the variables represent the meaning of the utterance. Both the structure of the finite-state network as well as the variable assignment are designed by a dialogue developer by means of a specialised regular grammar syntax [13].

Each state of the dialogue that expects user speech input has associated with it such a finite-state network. The network describes the possible utterances expected from the user at that particular point in the dialogue. The string of words recognised by the speech recognition component is passed to the natural language understanding module, which determines the particular path through the network represented by the recognised utterance. In order to ensure that such a path can always be found, our speech recogniser employs the same finite-state network to constrain its recognition search. Finally, the variable assignments made along this path are identified to extract the meaning of the utterance.

At this point one might ask whether it is possible to specify all possible user utterances by means of a finite-state network. Although this can not be guaranteed and there are generally an unlimited number of different ways in which a user may respond to the system, people tend to reply in predictable ways. In particular, they tend to reply using the words they have heard the system use in its most recent prompts [2, 7]. By carefully designing the system prompts in conjunction with the finite-state understanding grammars, it is possible for the dialogue developer to minimise the probability of a user's reply lying outside the specified set of allowable utterances. Iterative system refinement is achieved by repeatedly updating the understanding grammars after performing a set of user trials, so as to include unanticipated user responses. We have followed this approach in developing our hotel-reservation prototypes.

5. THE DIALOGUE CONTROLLER

Our dialogue controller is implemented as a state machine where each state fulfills a specific role in the dialogue. In particular, the following steps are executed sequentially in each state.

(a) Information is conveyed to the user by means of synthesised speech.

(b) The user's response to the system's speech in step (a) is captured by means of speech recognition.

(c) The recognised user utterance from step (b) is interpreted by means of natural language understanding.

(d) On the basis of information obtained from the user's utterance in step (c) the system chooses the most appropriate next dialogue state.

(e) Execution branches to the dialogue state determined in step (d).

Steps (b) and (c) are omitted in states that serve only to convey information to the user and do not require a response. Step (d) is trivial if the current dialogue state has just one possible successor.

Dialogue system development begins with the design of a state diagram as a solution to a particular problem, and the specification of steps (a)-(d) for each state.

5.1. *System prompts*

A **system prompt** is a speech utterance generated by the dialogue system, as required in step (a) above. In our architecture, every dialogue state has associated with it one or more of the following possible prompts.

- The Main prompt is generated when the state is first entered. Its purpose is to convey the function of the state to the user. The system utterances in the dialogue fragment used as illustration in section 1 are *Main* prompts.
- The Help prompt is generated when the user asks for help. It describes how the current state expects the user to respond. The user may ask for help at any point in the dialogue.

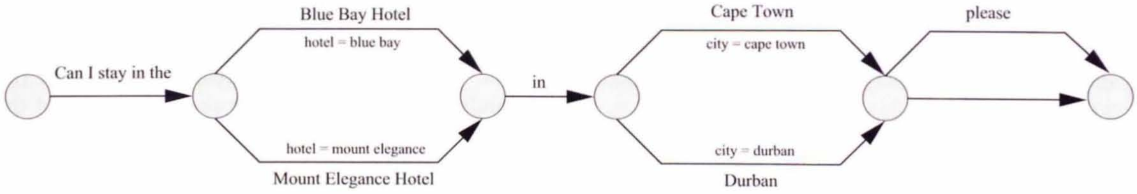

Figure 2: Example of finite state recognition network.

- The *ReEntry* prompt is generated when a state is entered for the second time. This happens for example when the user has specified an invalid information item and the system must prompt the user for that same information again. The *ReEntry* prompt is usually phrased differently from the *Main* prompt in order to avoid monotony and to ensure that unnecessary information is not repeated.

- The *Timeout1* and *Timeout2* prompts are generated when the system has waited for a user utterance for a predetermined period of time without detecting any speech. The *Timeout1* or *Timeout2* prompt is used depending on whether this has occurred for the first or for the second time in this state. The two prompts are phrased slightly differently in order to avoid monotony.

- The *Retry1* and *Retry2* prompts are generated when the system was unable to recognise the user's utterance (i.e. a speech recognition failure occurs). As before, *Retry1* and *Retry2* are phrased slightly differently in order to avoid monotony.

- The *Operator* prompt is generated when the user asks to be transferred to a human operator. Our system permits this at any point in the dialogue.

- The *Goodbye* prompt is generated when the user asks to quit the dialogue. Again, this may occur at any time.

- The *Back* prompt is generated as confirmation when the user asks to go back to the previous dialogue state.

- The *Repeat* prompt is generated when the user asks for the *Main* prompt to be repeated, possibly because he has not understood it the first time. The *Repeat* prompt is played, followed by a repetition of the *Main* prompt.

Only the *Main* prompt is compulsory. For example, if a state does not expect user input, the system proceeds to the next appropriate state immediately after generating the *Main* prompt.

Prompt design is a very important aspect of spoken dialogue development since it to a large degree determines the naturalness and clarity of the dialogue. Furthermore, carefully crafted main prompts will reduce the ambiguity of the user's reply and allow it to be anticipated more easily.

For example, the system prompt:

> "*What type of room would you like?*"

may be expected to elicit a wide variety of responses from the user because the type of reply that is expected is not clearly implied. If the prompt were rephrased as:

> "*Would you like a single, double or suite?*"

the user's reply would be much more constrained, most often explicitly containing the words "single", "double" or "suite".

The more constrained the user's replies are, the easier and more successful the natural language understanding process will generally be.

5.2. *Dialogue design tools*

We have produced a graphical tool with which to develop the spoken dialogue state machine. The states, their various prompts, and the inter-state branch logic may all be specified in a graphical environment. This greatly speeds the process of dialogue development.

6. SPEECH SYNTHESIS

The speech synthesis component is widely considered to be one of the most important factors influencing the user's perception of the system's overall quality. General text-to-speech (TTS) systems synthesise speech from unrestricted input text, and would seem to be an obvious choice. However all but the most sophisticated TTS systems fall well short of the voice quality required for commercial acceptance. Furthermore this type of speech synthesis is presently not viable in Afrikaans and Xhosa, since the linguistic resources required for inferring the phonetic and prosodic properties of the spoken utterance from the text do not yet exist. Such resources would for example include models that predict phoneme intonation and duration as well as lexical and syntactic stress by taking account of phonetic, syllabic, lexical and grammatical context.

We have traded the flexibility of TTS systems for the higher quality achievable using **limited-domain concatenative synthesis**. This is made possible by the highly constrained nature of the set of utterances that need to be synthesised for a given dialogue. In particular, the finite-state structure of our dialogues allows the entire set of possible sentences that may have to be produced to be determined in advance.

When system prompts consist of fixed sentences, they are termed *static* and can be "synthesised" by simple playback of a static recording. When sentences contain variable information, such as dates, digit-strings and money amounts, they are referred to as *dynamic*. For example, the following dynamic prompt is used to confirm the cost of the hotel booking (in English and Xhosa):

- So for <integer> nights, that's a total cost of <money>.
- Ngamagumbi <integer_ama> anebhedi enye axabisa i-<money> lilinye.

For dynamic prompts one could in principle enumerate and record all possible combinations. However this would require an extremely large number of recordings even for a relatively simple dialogue system. Instead we select a subset of sentences that contains every word in the dialogue in each of its contexts. We have chosen the context of a word to mean its successor word, or the following grammatical break, whichever comes first [15]. Grammatical breaks include phrase- or utterance-final positions in questions and statements. This strategy ensures that the recordings cover all words used by the dialogue in all required prosodic and cross-word phonetic contexts.

The sentences in our chosen subset were recorded and subsequently annotated with time-aligned phonetic transcriptions. These were obtained by means of forced alignment, a technique also used extensively for speech recognition. Since our recorded sets are typically quite small, and these automatic methods work better with larger data sets, a number of errors had to be corrected by hand. Our hotel reservation system required approximately 300 recordings to be made in each of the three languages. We have found that the careful design and annotation of the recorded utterances to a very large extent determines the quality of the synthesised voice.

Speech synthesis now proceeds by selecting and concatenating appropriate short waveform segments from this set of recordings [5, 8]. We employ the Festival [3, 4] open source speech synthesis engine and toolkit for this purpose. For each system prompt, this synthesiser constructs a network of possible waveform segments with which the desired speech can be generated. A Viterbi search through this network subsequently finds the optimal sequence of candidate segments. A Euclidean distance measure at segment edges gives an indication of how perceptible the concatenation will be. This distance is combined with the differences in pitch and energy in a weighted sum to give a measure of the quality of each concatenation point. The Viterbi search uses this measure to compare competing paths through the network. Waveform segments that already happen to be consecutive are assigned a concatenation cost of zero. This biases the algorithm to greedily select longer consecutive waveform sections from the recordings.

By annotating the recordings at a phonetic level, we allow the algorithm greater freedom in choosing concatenation points. However, valid concatenation points are restricted to candidates occurring in the same phoneme of the same word in the two source waveforms. This restriction ensures that prosody is best preserved during the concatenation process.

A further reduction in the number of required recordings can be achieved by separating common prefixes and suffixes from word stems. These fragments are in effect treated as separate words in the process outlined above. This was especially useful for Xhosa due to its conjunctive structure.

Since the synthesised speech consists of concatenated portions of recorded speech, the voice is very human-like, which cannot be said for many TTS systems. Furthermore, the constraint that concatenated waveform segments must originate from the correct word-level context preserves the natural intonation and rhythm. In this way the difficult problem of computing the correct prosody is circumvented. For example, digit strings are read with a natural tempo and intonation pattern, unlike many voice-mail and directory enquiry playback systems that employ simple context-insensitive concatenation of isolated digits. The disadvantage of constrained concatenative speech synthesis is that new recordings must be made and a new voice built for each dialogue and perhaps even for extensions to existing dialogues.

7. APPLICATION EXECUTION FLOW

The four managers described in the previous sections are initialised prior to a telephone call being accepted by the dialogue system. The subsequent interaction between the software components occurs in the following manner.

1. The telephony manager detects an incoming call and reports this to the system manager.
2. The system manager initialises and configures a dialogue controller, which in turn requests the telephony manager to accept the call.
3. The dialogue controller enters the first dialogue state. It requests the prompt generation manager to generate the speech waveform for that state's main prompt.
4. The prompt generation manager generates the required waveform and passes it to the dialogue controller.
5. The dialogue controller passes the waveform to the telephony manager and requests that it be played to the user.
6. The telephony manager plays the prompt to the user.
7. If no user speech is expected in the current dialogue state, the dialogue controller branches to step 10. If user speech is expected, the dialogue controller instructs the telephony manager to listen for user utterances.

8. The telephony manager informs the dialogue controller whether a user utterance was detected and if so relays it to the recognition manager.

9. The recognition manager recognizes and interprets the speech and sets appropriate variables to reflect this interpretation. It then passes these variables to the dialogue controller.

10. The dialogue controller uses the variables passed to it by the recognition manager (if speech was detected) as well as the information in its database to decide on the most appropriate next dialogue state. This state is entered and execution branches to step 4.

8. THE HOTEL RESERVATION DIALOGUE SYSTEM

We have chosen a hotel-reservation task to test and demonstrate our dialogue system technology. The system allows a user to negotiate a fictitious hotel booking by telephone.

We have adopted a **system-directed** dialogue strategy in our prototypes. This means that the system actively maintains the initiative during the dialogue, while the role of the user is reactive. In this way the natural language understanding process is simplified, since it is not necessary for the system to guess the user's intention. Rather, the user is assumed to react immediately to the most recent prompt.

The hotel-reservation system operates by successively prompting the user for information items such as the city, the name of the hotel, the arrival and departure dates, the type of room, and the user's credit card details. It maintains a database of hotel-specific information, from which it determines room availability and pricing, and in which it stores a new booking.

Three separate systems were developed, operating in Afrikaans, South African English and Xhosa respectively. The English system was developed first. Initial prototypes were iteratively refined by observing their performance and the reactions of users during Wizard-of-Oz tests [7] and subsequently during full system trials. In this way the quality of the system was gradually improved. In particular, the prompts were frequently re-crafted in order to reduce any ambiguity experienced by the user and thus make his or her replies more predictable, as explained in section 5.1. The Afrikaans and Xhosa systems both started out as direct translations of the English system. However substantial individual improvements and optimisations were then made to both in order to take language and cultural factors into account. This was found to be very important. In particular, the directly translated systems without language-specific customisation were found to perform poorly.

In order to ensure robustness to speech recognition errors without overly lengthening the dialogue, we have employed an **implicit confirmation** strategy wherever possible. Instead of verifying each information item with the user before proceeding, the information is incorporated into the next prompt. The user can then interrupt the system and object if he has been misunderstood. Failure to object is interpreted as a confirmation by the system. Due to the elimination of explicit confirmation states, the total length of the dialogue is strongly reduced. This leads to greater user satisfaction since the dialogue can be completed more quickly.

The same fairly simple dialogue structure employing 42 states was used for each of the three languages. A state flow diagram as well as brief description of each dialogue state is given in the appendix. In its current form our system is not intended for commercial application, but serves as an easily-understood example which is nevertheless sufficiently complex for us to test our technology as well as perform some comparative analysis across the three languages.

9. SYSTEM EVALUATION

Our best Afrikaans, English and Xhosa systems were tested by requesting a set of volunteers to call the system and make a hotel reservation. The volunteers consisted of University staff and students, and each was presented with an information sheet describing the operation of the system before being asked to place a call. The calls were subsequently analysed and the results are presented in the following.

Table 2 describes the overall performance of each system. It is evident that, although the average number of utterances made per call is approximately the same for all three systems, the average duration of calls made to the Xhosa system is much longer than for the Afrikaans and English systems. Xhosa users showed a tendency to use longer sentences in their replies than their Afrikaans and English counterparts.

The last row of Table 2 shows the number of users who experienced difficulty in using the system, lost patience and terminated their calls. Analysis of the call logs indicate that in approximately 20% of cases these problems were caused by loud and frequent background noises which lead to severe speech recognition errors. In the remaining cases failure was mainly due to persistently errorful recognition for the speaker in question.

System	English	Xhosa	Afrikaans
Total Calls	77	76	78
Average call duration (sec)	267	391	273
Recorded utterances	2326	2246	2326
Average utterances per call	30.2	29.5	29.8
Successful reservations	60 (78%)	53 (70%)	65 (83%)
Unsuccessful calls	17 (22%)	23 (30%)	13 (17%)

Table 2: Overall system performance during user trials.

Table 3 summarises the performance of the three systems on a per-state basis, showing the total number of times each dialogue state was entered as well as the recognition performance in each. The appendix presents a brief description of each dialogue state, as well as a state flow diagram. All user utterances collected during the usability tests were later marked as *exactly correct, conceptually correct, incorrect* or *noise* depending on what the recording of the

State	English			Xhosa			Afrikaans		
name	#Utts	Exact	Concept	#Utts	Exact	Concept	#Utts	Exact	Concept
ArrivalDate	130	44%	57%	126	22%	43%	109	72%	77%
CardExpiry	134	38%	48%	91	39%	43%	82	63%	65%
CardType	114	62%	76%	67	80%	86%	83	77%	83%
Change	12	92%	92%	2	100%	100%	6	83%	100%
City	136	78%	85%	155	82%	92%	103	96%	97%
ConfirmGoodbye	51	90%	90%	8	100%	100%	8	100%	100%
ConfirmOperator	38	84%	92%	42	57%	86%	60	75%	82%
DepartureDate	155	65%	71%	260	30%	40%	141	63%	73%
DepartureDouble	100	74%	90%	120	42%	60%	110	54%	61%
DoubleConfirm	99	67%	86%	91	61%	88%	109	58%	65%
DoubleRooms	57	91%	97%	125	6%	23%	87	92%	92%
End	71	71%	83%	72	80%	89%	69	89%	96%
GlobalConfirm	91	96%	96%	73	78%	81%	95	97%	99%
Hotel	124	68%	83%	161	42%	62%	145	37%	59%
HotelInfo	84	82%	94%	93	80%	88%	95	89%	93%
Introduction	124	74%	78%	73	67%	77%	105	83%	87%
Restart	6	100%	100%	2	50%	50%	10	90%	90%
RoomsConfirm	73	46%	95%	63	90%	97%	95	72%	86%
SecurityCode	98	59%	61%	66	66%	69%	70	79%	79%
SingleRooms	58	82%	86%	22	22%	27%	80	88%	90%
ccCard1	95	60%	61%	81	41%	58%	95	64%	64%
ccCard2	131	62%	66%	79	54%	65%	121	52%	52%
ccCard3	126	66%	71%	75	50%	60%	106	57%	59%
ccCard4	121	54%	58%	75	50%	61%	113	57%	58%
ccCardConfirm	86	59%	95%	75	62%	80%	100	75%	88%
OVERALL	2314	66.7%	76.4%	2097	52.0%	64.2%	2197	70.8%	76.0%

Table 3: Per-state performance during usability testing.

user's utterance contained and whether it was recognized correctly. *Exactly correct* indicates that the speech recognition result corresponded exactly to the words uttered by the user. When the recognition result did not correspond exactly to the user's utterance but nevertheless had the same meaning, it was marked *conceptually correct*. For example, while a user had said:

> *"I'd like to stay at the Mount Elegance"*

our system transcribed this as

> *"I'd like to stay in the Mount Elegance"*.

Although a recognition error has occurred, this example would have been tagged as conceptually correct. This better reflects overall system performance, since not all recognition errors lead to conceptual errors and therefore did not disrupt the dialogue. Table 3 indicates the percentage of times the system was exactly and conceptually correct.

From Table 3 we see that overall the English system performed best with the Afrikaans system in second place and the Xhosa system in third. This mirrors the performance of the acoustic models as reported in section 3. However it was also found during the dialogue design process that the English and Afrikaans speakers tended to respond in a more predictable fashion to the system prompts. In particular, although the reply to a question can always be phrased in a large number of ways, English and Afrikaans speakers tended to choose similar ways of doing so respectively. As pointed out in section 5.1 this makes it easier to design the recognition and understanding grammars for each state. Xhosa users, on the other hand, tended to be more individual in how they responded to a question. In particular, it was found that Xhosa users often repeat information already confirmed in previous dialogue states. For example, the "Hotel" state prompts only for the name of the hotel. However many Xhosa users included the name of the city in their reply although this had already been determined in the previous state and had also been repeated in the "Hotel" state's main prompt as implicit confirmation.

In general, when asked to supply dates, times, numbers and amounts, Xhosa speakers may elect to reply in English since this is often a shorter and hence more convenient alternative. For example, the amount:

R2353.20

is most often spoken simply as:

> *"Two thousand three hundred and fifty three rand and twenty cents"*.

However it could also be spoken as:

> *"Amawaka amabini anamakhulu amathathu namashumi amahlanu anesithathu eerandi kunye neesenti ezingamashumi amabini"*,

meaning literally:

> *"Thousands-that-are-two and hundreds-that-are-three and tens-that-are-five and three of rands and cents of tens-that-are-two"*.

Although Xhosa speakers almost always choose English to speak longer amounts and numbers such as used in the above example, this is not true for shorter numbers like those occurring in dates or those used to specify the number of hotel rooms. Hence the "ArrivalDate", "DepartureDate", "SingleRooms" and "DoubleRooms" states, for example, must accept replies in both Xhosa and English. Consequently these states have more complex recognition and understanding grammars for the Xhosa dialogue system than they do for the Afrikaans and English systems. This in turn has led to poorer Xhosa per-state recognition performance, as reflected in Table 3.

10. SUMMARY AND CONCLUSIONS

This paper has described the architecture and performance of a spoken dialogue system operating in Afrikaans, South African English and Xhosa. It has been found that the same basic dialogue structure can be used successfully in each of the three languages. In particular a spoken dialogue system operating in Xhosa has been demonstrated to be feasible for the first time.

Experience gathered during system development and the subsequent quantitive comparison of the systems' performance indicates that the responses of Xhosa users tend to be longer and more varied that those of Afrikaans and English users. Furthermore, Xhosa speakers may switch to English when providing numerical information such as dates, times, numbers and amounts. This means that the careful design of the dialogue structure and especially the state prompts and grammars is particularly critical for Xhosa systems. Whether this can be concluded for other African languages is the subject of current research. In particular, we will place further focus on the importance and effect of bilingualism and code-mixing in the context of South African spoken dialogue systems. Multilingual speech recognition as well as language and accent identification are two particular aspects that are relevant in this regard.

11. ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive comments.

12. REFERENCES

[1] E. Barnard, A. Halberstadt, C. Kotelly, and M. Phillips. "A consistent approach to designing spoken-dialog systems". In *Proc. ASRU*, pages 363–368, Keystone, Colorado, USA, 1999.

[2] N.O. Bernsen, L. Dybkjaer, and H. Dybkjaer. "Principles for the design of cooperative spoken human-machine dialogue". In *Proc. ICSLP*, pages 729–732, Philadelphia, USA, 1996.

[3] A.W. Black and K.A. Lenzo. "Limited domain synthesis". In *Proc. ICSLP*, Beijing, China, 2000.

[4] A.W. Black and K.A. Lenzo. http://www.festvox.org/, June 2004.

[5] A.W. Black and P. Taylor. "Automatically clustering similar units for unit selection in speech synthesis". In *Proc. Eurospeech*, Rhodes, Greece, 1997.

[6] R. Carlson and S. Hunnicutt. "Generic and domain-specific aspects of the waxholm NLP and dialog modules". In *Proc. ICSLP*, pages 677–680, Philadelphia, USA, 1996.

[7] H. Dybkjaer, N.O. Bernsen, and L. Dybkjaer. "Wizard-of-Oz and the trade-off between naturalness and recogniser constraints". In *Proc. Eurospeech*, pages 947–950, Berlin, Germany, 1993.

[8] A. Hunt and A.W. Black. "Unit selection in a speech synthesis system using a large speech corpus". In *Proc. ICASSP*, Atlanta, USA, 1996.

[9] D. Jurafsky, C. Wooters, G. Tajchman, J. Segal, A. Stolcke, E. Fosler, and N. Morgan. "The Berkeley Restaurant Project". In *Proc. ICSLP*, pages 2139–2142, Yokohama, Japan, 1994.

[10] L.F. Lamel and J.L. Gauvain. "High performance speaker-independent phone recognition using CDHMM". In *Proc. Eurospeech*, pages 121–124, Berlin, Germany, 1993.

[11] M.F. McTear. "Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit". In *Proc. ICSLP*, Sydney, Australia, 1998.

[12] T.R. Niesler and P.H. Louw. "Comparative phonetic analysis and phoneme recognition for Afrikaans, English and Xhosa using the African Speech Technology telephone speech databases". *South African Computer Journal*, 32:3–12, June 2004.

[13] T.R. Niesler and J.C. Roux. "Natural language understanding in the DACST-AST dialogue system". In *Proceedings of the twelfth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, pages 134–136, Franchhoek, South Africa, November 2001.

[14] J.C. Roux, P.H. Louw, and T.R. Niesler. "The African Speech Technology project: An assessment". In *Proc. LREC*, volume 1, pages 93–96, Lisbon, Portugal, 1998.

[15] J. P. H. van Santen and A. L. Buchsbaum. "Methods for optimal text selection". In *Proc. Eurospeech*, pages 553–556, Rhodes, Greece, 1997.

[16] S.J. Young. *Token passing, a simple conceptual model for connected speech recognition systems*. Technical Report TR38, Cambridge University, 1989.

[17] S.J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK book, version 2.2*. Entropic, 1999.

APPENDIX A

This appendix describes the overall structure and working of the hotel-reservation dialogue. A state flow diagram for the dialogue is shown in in Figure 3, while the following gives a brief description of the function of each state.

- Welcome - This is the entry point of the dialogue. A short welcoming prompt is played to the user.
- Introduction - This state immediately follows the *Welcome* state and outlines the most important aspects of the system.
- Tips - Here the user is given more detailed information on how to use the system. The user is asked whether he would like to hear this information in the *Introduction* state.
- Start - The user is informed that the booking process is about to begin and that he should have all the necessary information at hand.
- City - The user is supplied with a list of cities in which hotels are available, and is asked to name the one in which he would like to stay.
- Hotel - The user is supplied with a list of available hotels in the specified city. The name of the city is included in the prompt as implicit confirmation.
- InvalidHotel - The user is informed that the hotel he has chosen is not in the city he has chosen.
- HotelInfo - The user is asked whether he would like further information regarding the selected hotel. The name of the hotel is included in the prompt as implicit confirmation.
- InfoOnHotel - Further information on the selected hotel is played to the user.
- ArrivalDate - The date of arrival is acquired from the user.
- InvalidArrDate - The user is informed that an invalid arrival date has been specified. This occurs for example when the date does not exist (such as the thirty first of February) or when it is already in the past.
- DepartureDate - The departure date is acquired from the user. The arrival date is included in the prompt as implicit confirmation.
- InvalidDepDate - The user is informed that an invalid departure date has been specified. This occurs when the date does not exist, or when it precedes the arrival date.
- DepartureDouble - The user is asked whether he would like any double rooms. The departure date is included in the prompt as implicit confirmation.
- DoubleRooms - The user is asked how many double rooms he would like to book.
- DoubleConfirm - The user is asked whether he would like any single rooms. The number of double rooms is included in the prompt as implicit confirmation.
- SingleRooms - The user is asked how many single rooms he would like to book.
- GlobalConfirmation - The user is informed that the system is checking for availability of the rooms. The number of single rooms is included in the prompt as implicit confirmation.
- ZeroRooms - The user is informed that he has chosen to book a total of zero rooms.
- NoRooms - The user is informed that the desired booking is not possible due to insufficient room availability.
- Change - The user is asked whether he would like to amend his booking.
- RoomsConfirm - The details of the booking are confirmed explicitly. The total length of stay and the cost are stated and the user is asked whether this is satisfactory.
- CardNoIntro - The user is notified that the credit card details will be asked for next.
- ccCard1,2,3,4 - These four states prompt the user to state the first, second, third and fourth group of four digits of the credit card number respectively. The 16-digit number was broken up into four groups of four in order to aid error recovery.
- ccCardConfirm - The credit card number is confirmed explicitly.
- RetryccCard - This state is entered when the user notifies the system that the credit card number has not been recognised correctly.
- CardHolder - The user is asked to specify the name of the card-holder. This speech item is logged only (no speech recognition is attempted).
- SecurityCode - The user is asked to read the credit card's security code.
- InvalidSecCode - The user is informed that he has specified an invalid security code.
- CardType - The user is asked to specify the type of credit card (e.g. Mastercard).
- CardExpiry - The user is asked to specify the expiry date of the credit card.
- InvalidExpiry - The user is informed that an invalid expiry date has been specified.
- ExpiredCard - The user is informed that his credit card has already expired.
- ReservationNumber - The user is informed that the booking has been completed successfully and is given a reservation number.
- End - The user is asked whether he would like to make another booking.
- Goodbye - This is the final state of the dialogue. The user is thanked and greeted, after which the call is terminated.

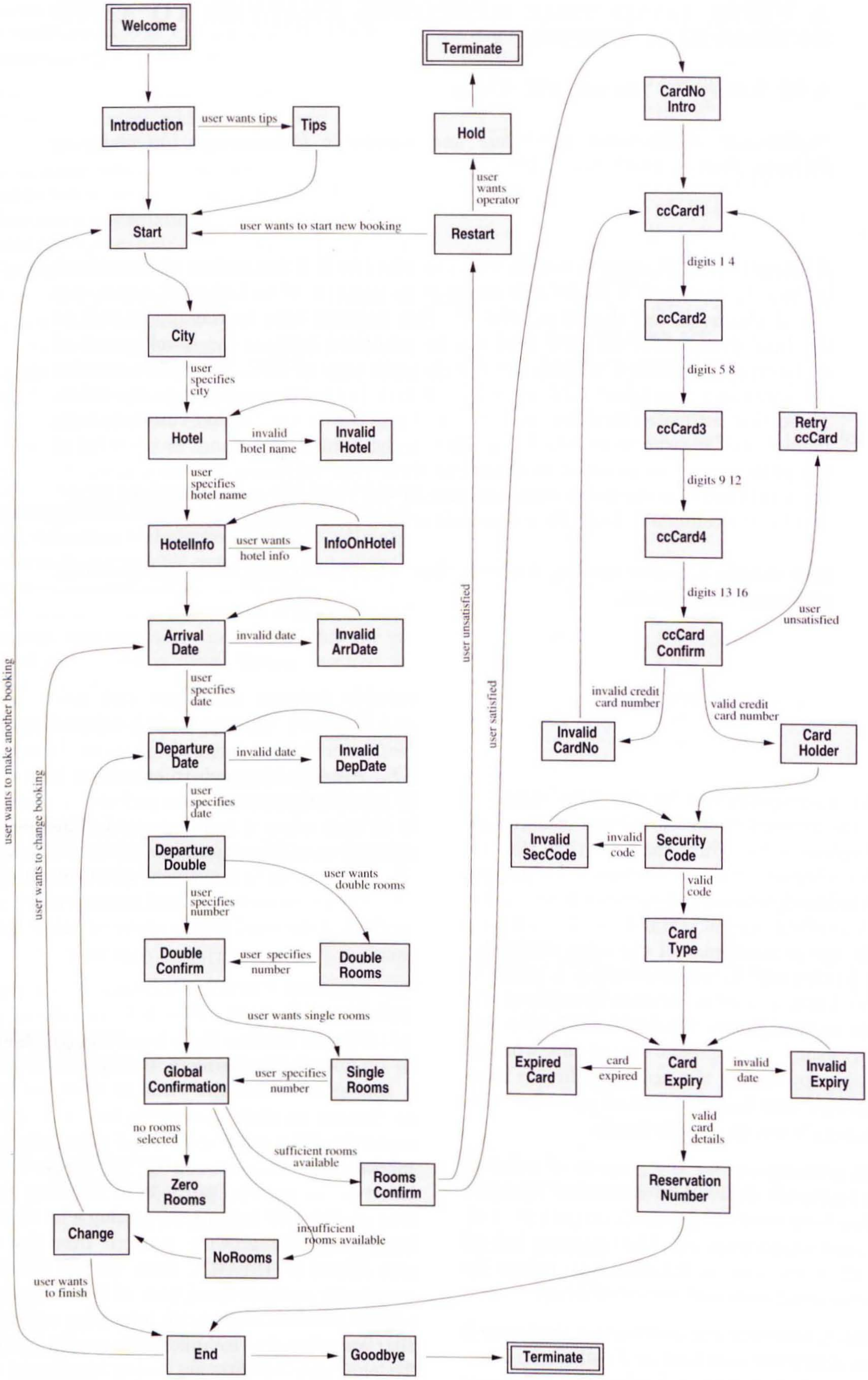

Figure 3: Dialogue state flow diagram of the Hotel Reservation System.