INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 2, 121–131 (1997)

# A Spoken Dialog System for the Dutch Public Transport Information Service

HELMER STRIK, ALBERT RUSSEL, HENK VAN DEN HEUVEL, CATIA CUCCHIARINI AND LOU BOVES

*Department of Language and Speech, University of Nijmegen, The Netherlands*

strik@let.kun.nl

*Received June 10, 1996; Accepted March 18, 1997*

**Abstract.** In the Netherlands there is a nationwide premium rate telephone number that can be dialed to obtain information about various forms of public transport. In 1996 this number was called more than twelve million times. Human operators managed to handle only about nine million of these calls. In order to answer more of these calls, a spoken dialog system was developed to automate part of this service. The automation component concerns information about journeys between two train stations.

The starting point of our research was an existing German information system. This system was ported to Dutch. A bootstrapping method was used to collect the data, which in turn were used to improve the system itself.

**Keywords:** spoken dialog systems, public transport information, continuous speech recognition, natural language processing

## 1. Introduction

Information on public transport in the Netherlands can be obtained by dialing a well-known telephone number. This service is provided by trained operators, who are available seven days a week, from 6 am to midnight. The operator service uses a data base that contains the schedule information of all public transport companies in the country. Therefore, the operators can provide the caller with information about the various forms of public transport in the Netherlands: train, metro, tram, bus, and interliner (inter-regional bus lines). Since the service became available, transport companies no longer answer questions about schedules; callers are always referred to the central service.

Callers are asked to specify their origin and destination locations. Origin and destination are often addresses, specified as street name and house number. Alternatively, 'special locations' like train stations, hospital names, museums, and other major venues can be given for origin and destination. In addition, date and time of travel must be provided. Finally, the caller must say whether the specified time is departure or arrival time. A data base query is made with this input information which will return the schedule that will bring the caller from origin to destination in the shortest possible time. Analysis of transcripts of calls handled by the operators shows that determining the 'best' schedule can require a number of queries and answers.

In 1996, slightly more than twelve million calls were counted by the automatic call distribution system. Of these only nine million were processed by the operators; the remaining three million calls were abandoned before an operator became available. Call statistics collected by the operators show that 45% of the queries pertain to journeys between two train stations. Also, it appears that the dialogs needed to identify these departure and destination locations tend to be straightforward, i.e., the caller usually provides unambiguous information to the operator. Moreover, only in very exceptional cases is there a follow-on discussion about alternative, more convenient schedules. Therefore, it seems both worthwhile and feasible to try and automate the processing of the queries that are limited to travel between two train stations: the application is a 'straightforward information service' and it covers a substantial portion of the demand. If, for example, half of these queries could be handled by a machine, this would correspond roughly to the three million calls that are presently aborted due to excessive waiting times in the queue for the operator service.

In this paper we describe our attempts to localize an existing prototype Spoken Dialog System (SDS) that handles train time table information for the German Railways, in order to make it suitable for the Dutch public transport information service. A general description of the system is given in Section 2. In Section 3 we describe the bootstrapping method that was used to port the SDS to Dutch. Section 4 deals with the integration of the SDS in the operational service. Finally, a discussion and the conclusions are presented in Section 5.

## 2. General Description of the System

The starting point of our research was the prototype of an automatic inquiry system developed by Philips Research Labs (Aachen, Germany), which can give information about the schedules of the German railways. In this section only a short description of the general properties of the system is given. Specific properties of the Dutch system are presented in the next section (see also, Strik et al., 1996), while further details about the system can be found in (Oerder and Ney, 1993; Steinbiss et al., 1993; Aust et al., 1994; Ney and Aubert, 1994; Oerder and Aust, 1994; Steinbiss et al., 1994; Aust et al., 1995; Steinbiss et al., 1995). Conceptually, the Spoken Dialog System (SDS) consists of four parts (in addition to the telephone interface and the data base with public transport information):

(1) Continuous Speech Recognition (CSR) module,
(2) Natural Language Processing (NLP) module,
(3) Dialog Management (DM) module, and
(4) Text-To-Speech (TTS) module.

The system was connected to an ISDN line. Therefore, the input signals consist of 8 kHz 8 bit A-law coded samples. Feature extraction is done every 10 ms for frames with a width of 16 ms (Steinbiss et al., 1993, 1995). The first step in feature analysis is an FFT analysis to calculate the spectrum. Next, the energy in 14 mel-scaled filter bands between 350 and 3400 Hz is calculated. Apart from these 14 filterbank coefficients the 14 delta coefficients, log energy, and slope and curvature of the energy are also used. This makes a total of 31 feature coefficients.

The CSR uses acoustic models (HMMs), language models (unigram and bigram), and a lexicon. The lexicon contains orthographic and phonemic transcriptions of the words to be recognized. The continuous density HMMs consist of three segments of two identical states, one of which can be skipped. The output of the CSR module, and thus the input to the NLP module, is a word graph (Oerder and Ney, 1993; Ney and Aubert, 1994).

The NLP's task is to decide which path through the word graph has to be chosen. The NLP does not choose this path by looking at the acoustic likelihood of the path alone. It also uses application-specific knowledge in the form of a concept bigram and syntactic unit counts. The goal of the NLP module is not to find a parse for the complete utterance, but to look for sequences of concepts in the word graph. The concepts it looks for are defined in a stochastic attributed context-free grammar (ACFG) that describes the utterances which must be understood. For instance, the entry "<departure_station> ::= (121) from <station_name>" is a part of the ACFG and is one of the many entries that define the concept <departure_station>. It denotes that if a path through the word graph exists with e.g., the utterance "from Amsterdam" in this, it should be interpreted as a statement indicating that the departure station is Amsterdam. Of course, the same holds for the names of other cities present in the lexicon. A similar definition exists for the related concept <arrival_station>: "<arrival_station> ::= (248) to <station_name>". The numbers (121) and (248) are the syntactic unit counts for these concept definitions. They state that these syntactic units occurred 121 and 248 times, respectively, in the corpus on which the NLP was trained. The concept bigram in the NLP describes the frequency of occurrence of ordered pairs of concepts, just as a standard language model bigram does for ordered pairs of words. The combination of concept bigram values, syntactic unit counts, and the acoustic likelihood of the phonemes decides which path through the word graph is most likely in this application (Aust et al., 1995).

The DM gathers all the necessary information to perform a data base query, by asking specific questions to the caller. The DM needs to know the departure and arrival station, the departure or arrival time and the date on which the caller wants to travel. The opening question of the DM is "From which railway station to which railway station would you like to travel?". If the caller answers "I want to travel from Nijmegen to Amsterdam tomorrow" the NLP sets the values for departure_station := Nijmegen, arrival_station := Amsterdam, and date := tomorrow. The DM always gives feedback about recognized concepts, thus making it possible for the caller to correct errors (see Section 3.1.3). If two successive attempts to obtain an interpretable answer to a question have failed, the DM gives up. An apology message is played to the caller, advising her or him to call the operator service. The caller is then disconnected.

The information found in the data base (and all other responses mentioned above) is presented to the caller by means of speech synthesis. Language generation is limited to the concatenation of fixed phrases or by inserting the right words in open slots in carrier phrases. Speech synthesis is accomplished by concatenating pre-recorded phrases and words spoken by a female speaker.

## 3. Building a Dutch SDS

In order to build and train an SDS for a certain application, considerable data are needed. Wizard-of-Oz scenarios are often used to collect these data. However, within the framework of the current research a different approach was chosen, which will be called the bootstrapping method (Aust et al., 1994; Oerder and Aust, 1994). By taking the existing German prototype, a bootstrap version of the SDS was made without using application-specific speech data. To develop this version we could rely on our knowledge of the application and of the two languages, German and Dutch. To train the acoustic models, an existing speech data base was used. Finally, data collected by means of a keyboard version of the system (i.e., a version without speech interfaces) were employed to initialize the language model and the natural language processing module.

This bootstrap version was placed in the PSTN (Public Switched Telephone Network) and was first used by a limited group of users. In this way application-specific data were collected. In turn these data were used to gradually improve the system.

The bootstrapping method adopted to collect the data and to develop the SDS consists of the following five stages:

(1) make a bootstrap version of the SDS with available data;
(2) ask a limited group of people to use this system, and store the dialogs;
(3) use the recorded data to improve the SDS;
(4) gradually increase the data and the number of users;
(5) repeat steps 2, 3, and 4 until the system works satisfactorily.

### *3.1. The First Version of the SDS*

In Section 2 we provided a short general description of the system. A first version of the SDS was obtained by customizing this German system for Dutch. How this was done is described in the present section.

***3.1.1. CSR.*** The CSR component of the first version of the SDS was trained by using part of the Polyphone data base (Damhuis et al., 1994; den Os et al., 1995). This corpus is recorded over the telephone and consists of read and (semi-)spontaneous speech of 5000 subjects. For each speaker 50 items are available. Five of these 50 items are the so-called phonetically rich sentences, which contain all phonemes of Dutch at least once, while the more frequent phonemes occur more often. The five phonetically rich sentences of 500 subjects (i.e., 2500 utterances in total) were used to train the acoustic models of the first version of the CSR component. In the current Dutch version 38 monophones are used. Thirty-five of these models represent phonemes of Dutch, two represent allophones of /l/ and /r/, and one model is used for all non-speech sounds.

Note that the data used to train the first version of the CSR are not application-specific. First of all, the data consist of read speech, and not spontaneous speech as in the intended application. Furthermore, the phonetically rich sentences of the Polyphone data base were selected from newspaper texts. Therefore, none of them are related to travels between two train stations.

The phonemic forms in the lexicon were taken from three different sources. Station names came from the ONOMASTICA data base (Konst and Boves, 1994), while the lemma forms of other words were taken from the CELEX data base (Baayen et al., 1993). The phonemic forms of words that were not found in these two data bases were generated by means of our grapheme-to-phoneme converter (Kerkhoff et al., 1984). Postprocessing was used to resolve systematic differences in the phonemic forms obtained from these three sources, i.e., to make the phonemic representations in the lexicon more homogeneous. For 42 words multiple pronunciations are included in the lexicon: 2 phonemic transcriptions for 40 words, and 3 phonemic transcriptions for 2 words. Of these 42 words 26 are station names, 10 are digits, and 6 are greetings. For all other words one phonemic transcription is present in the lexicon.

*3.1.2. NLP.* Since German and Dutch are rather similar from a syntactic point of view, for some parts of the NLP it was possible to make a direct translation from German to Dutch. However, in many other cases, such as time and date expressions, things appeared to be more complicated. First of all, each language has its own expressions for special days. For instance in Dutch we have "koninginnedag" (birthday of the queen, 30 April), which does not exist in German. Since it is common to say e.g., "de dag voor/na koninginnedag" (literally: the day before/after queen's day), the system had to be taught to recognize expressions of this kind.

A second problem with the interpretation of time is related to the ambiguity of certain time expressions. The word "tomorrow" uttered in the small hours is a case in point. In order to interpret time expressions, the system uses the internal clock of the computer on which the NLP software runs. If the caller says "tomorrow", the NLP adds 1 to the date of the internal clock. However, if the person calls at 00:10 and says "tomorrow", (s)he probably means "today", or, to be more precise, the current date of the internal clock. Therefore, this latter interpretation is adopted in the Dutch system.

Another date-related problem had to do with the fact that in the Dutch railway system there is a discrepancy between calendar days and 'schedule' days. A schedule day—which corresponds to the duration of the validity of tickets—starts at 04:00 and continues until 03:59 of the next calendar day. The data base that provides the train timetable information in the Dutch dialog system adheres to this definition of schedule day. Therefore, all date-related expressions that are somehow connected to the time span between 00:00 and 04:00 had to be reconsidered. For instance, if a caller says "I want to travel today" at 02:00, the German system should interpret 'today' as the date of the internal clock, whereas the Dutch system should interpret 'today' as one day before the calendar date.

Furthermore, the interpretation of time intervals had to be adjusted. The system allows queries for connections in a certain time interval. The start and end time of an interval must always be on the same 'day'. If a 'day' covers the interval from 04:00 to 03:59, the conventional numeric order is not preserved. When a caller says "I want to arrive between six and three", the present implementation allows for three interpretations, viz. 06:00–15:00, 06:00–03:00 and 18:00–03:00. In the original implementation only 06:00–15:00 would have been valid. Which of the three above-mentioned interpretations is chosen depends on the time of day when the query is received.

We were convinced that introspection alone would not derive all the expressions Dutch people could use in order to get information about public transport. At the same time, we did not have a large data base available to extract all possible expressions. Therefore, an alternative approach was adopted. A preliminary version of the grammar was made by translating and, where necessary, adapting the German grammar. This part of the SDS was then tested independently of the speech interfaces by using a keyboard version of the dialog system. A limited group of motivated people (about twenty) were asked to log in to the system and type their questions on a keyboard. The exact number of subjects who did participate is not known because the test was anonymous. The replies from the system appeared on the screen. Because people are likely to formulate their questions differently when they speak or type, the users were instructed to try to express themselves as they would if they were speaking.

In this way we were able to test the grammar and to gather some text material that could be used to initialize the language model. In total 243 dialogs were obtained, each consisting of at least 1 and as many as 6 queries. It turned out that the sessions of the users with this version of the NLP were extremely useful. On the basis of the log-files, many adjustments were made to the system. For example, in the original German grammar there are 18 ways to give an affirmative answer and 7 ways to give a negative answer. On the basis of the log-files 34 affirmative answers and 18 negative answers were defined for Dutch.

*3.1.3. DM.* For the bootstrap version of the system the German DM was translated literally into Dutch. Some adaptations appeared to be necessary, though. The interface to the public transport data base had to be modified. Furthermore, some changes were required in the responses of the system. For instance, the German system mentions train numbers because these appear to be important for the caller. However, this piece of information is irrelevant in the Netherlands (people never refer to the train number) and was therefore excluded from the replies of the Dutch system. In order to illustrate the way the DM works, a sample dialog is presented here:

System 1: "From which railway station to which railway station would you like to travel?"
Caller 1: "I would like to go to Arnhem"
System 2: "From which station would you like to leave for Haarlem?"
Caller 2: "I don't want to go to Haarlem but to Arnhem"
System 3: "From which station would you like to leave for Arnhem?"
Caller 3: "Amsterdam"
System 4: "When do you want to leave from Amsterdam?"
Caller 4: "I want to arrive in Arnhem tomorrow before ten o'clock"
System 5: "Do you want to arrive tomorrow before ten o'clock am?"
Caller 5: "Yes, that's right"

At this point in the dialog all the information needed to start a data base query is available.

In its responses the system asks for the information it still needs to fill the empty slots. For instance, in responses 3 and 4 the system asks for the place and time of departure, respectively. Furthermore, the system verifies all the recognized concepts. This is done by giving the caller explicit (e.g., response 5) or implicit (e.g., responses 2 to 4) feedback. The caller can then disconfirm erroneous items and replace them with correct information (e.g., utterance 2 of the caller). If the caller does not rectify the concepts recognized by the system (e.g., utterances 3 to 5 of the caller) then these items are considered to be correct and are frozen. This means that the caller can no longer change their values (Aust et al., 1994). If the DM has all the information it needs, it will do the data base query and report to the caller what connection(s) it found.

*3.1.4. TTS.* Many adaptations had to be made to the speech output module of the system, because only the general approach from the German prototype could be copied. In this approach stored speech units are concatenated to form the response of the system. An inventory was made which showed that 743 speech fragments were needed to generate the responses. These fragments were extracted from utterances spoken by a female speaker.

For the digits 1 to 59 two versions were recorded, i.e., an utterance-final and an utterance-internal version. In this way the intelligibility of the digits was improved. The zero can never occur in utterance-final position, which is why only an utterance-internal version was stored. This makes a total of 119 stored units for the digits. In addition there are speech units for the 7 weekdays, the 12 months, and 427 station names. The other 178 fragments are needed to make complete utterances (see e.g., the example dialog above). The responses of the system are made audible by concatenating these stored speech fragments.

### 3.2. *Improving the SDS*

The first version of the SDS was put in the PSTN in December 1995. The CSR module in this version was trained with DB0, i.e., the 2500 Polyphone utterances. Eighty employees of KPN (Royal Dutch PTT) received the telephone number of this system. The main criterion used to select them was that they had some experience with spoken dialog systems. However, they were not familiar with this specific SDS. They were requested to call it regularly, and their dialogs were recorded. In this way the data bases DB1 to DB3 in Table 1 were collected. Next, this telephone number was made known to 1200 other employees of KPN, and data bases DB4 to DB7 were recorded. The data bases in Table 1 are built up incrementally, which means that DB2 is a superset of DB1, DB3 of DB2, etc.

Each utterance was orthographically transcribed. Out-of-vocabulary (OOV) words were detected automatically from the transcriptions. In this way words containing typing errors were also found and manually corrected. The OOV words were phonematized by hand and added to the training lexicon, so as to use all the collected data for training the system. However, not all new words were added to the recognition lexicon. Only those words considered to be relevant to the application were included in the recognition lexicon. Consequently, the size of the recognition lexicon gradually increased. Currently the recognition lexicon contains 1115 words.

Since words not present in the recognition lexicon can never be recognized correctly, it is important that the number of OOV words not be too large. To ensure this, we use the relative number of OOV words, which is calculated by dividing the number of OOV words for a data base by the total number of words in that data base. The relative number of OOV words as a function of the number of words in the data bases is shown in Fig. 1. It can be observed that the relative number

*Table 1.* Data bases used during the development of the SDS. Presented in the columns are the name of the data base, the number of utterances, the source of the data, the total duration of all speech signals, the total number of words, and the total number of distinct words.

| Data bases | Utterances | Source | Duration | No. of words | No. of distinct words |
|---|---|---|---|---|---|
| DB0 | 2500 | Polyphone | 282 min. | 28516 | 5951 |
| DB1 | 1301 | Application | 41 min. | 4575 | 370 |
| DB2 | 5496 | Application | 227 min. | 22167 | 678 |
| DB3 | 6401 | Aapplication | 275 min. | 23279 | 683 |
| DB4 | 8000 | Application | 355 min. | 28197 | 785 |
| DB5 | 10003 | Application | 440 min. | 34587 | 845 |
| DB6 | 21288 | Application | 921 min. | 70773 | 1069 |
| DB7 | 32971 | Application | 1406 min. | 106880 | 1226 |

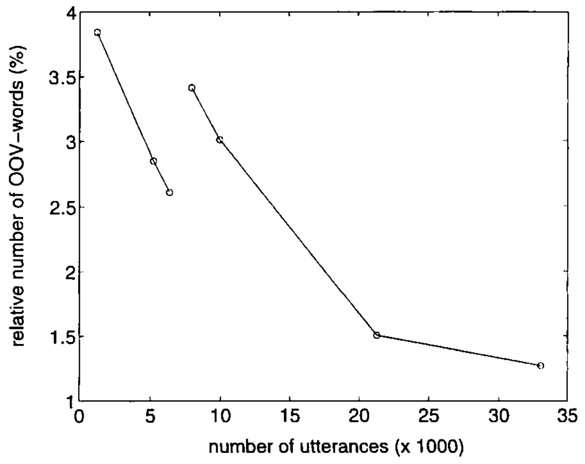

*Figure 1.* The relative number of out-of-vocabulary words as a function of the number of utterances in the database.

of OOV words is small. Apparently, we succeeded in making a bootstrap lexicon that contains most of the words used.

In Fig. 1 one can also see that the relative number of OOV words decreases as the number of utterances increases from 1301 to 6401. In the beginning a fair number of OOV words are found. However, as the same group of people is likely to use more or less the same words to ask for information, the number of unknown words gradually decreases. After DB3 (6401 utterances) had been recorded, the telephone number of the system was made available to a larger group of people. It is conceivable that new people will use new words. As a matter of fact, the relative number of OOV words turns out to increase first and to decrease again later on (see Fig. 1).

Whenever a sufficient amount of new data was collected, the lexicon was updated and language and phoneme models were trained again. The new models were compared to the old models (as will be described below), and those which performed best were chosen. In the on-line system the old models and the lexicons were replaced by the better ones.

In the early versions of the system we detected some syntactic constructions that were sometimes used by the callers, but were not handled correctly by the NLP. To improve the NLP, these syntactic constructions were added to the NLP's context free grammar. Furthermore, the NLP was trained with the same data used to train the language model (the bigram). During the training of the NLP the concept bigram model is constructed and the number of occurrences of syntactic units in the context free grammar is counted and stored in the NLP. As described above (see Section 2), the concept bigram and the syntactic unit counts are used in deciding which parse of the word graph is chosen.

Although the first bootstrap version of the system was quite useful as a tool for data acquisition, tests performed recently show that some changes at the ergonomic level are required. For instance, the concatenation synthesis should be improved, information about complex journeys should be split into smaller chunks, and the caller should be able to interrupt the machine (barge-in capability). Some of these improvements in the DM module will be addressed in the near future.

### 3.3. *Evaluating the Performance of the CSR Module*

Part of the data collected with the on-line SDS was kept apart as a test data base. Since we did not have much data in the beginning, the test data base had to be small: 500 utterances were randomly selected from the recorded dialogs. As a result, some of the utterances of a call are in the test data base, while the remaining part of the same call (and thus the same speaker) is in the training set. In total the test data base contains utterances from 465 different dialogs. The total number of words and characters in this test data base is 1674 and 9696, respectively. The total number of characters (or graphemes) can be used as a rough estimate of the total number of phonemes in the test data base.

The performance of the CSR module was evaluated for the whole word graph (WG) and for the best sentence (BS) obtained from this word graph. Both for the word graph and for the best sentence, word-error rate (WER) and sentence-error rate (SER) were calculated. In total this yields four measures that can be used for evaluation: WG-WER, WG-SER, BS-WER, and BS-SER. Because not everyone uses the same definition of WER it should be noted here that for calculation of the WER, insertions also counted as errors. Error rates for the word graph are obtained by searching the optimal path in the wordgraph, i.e., the path that matches the spoken utterance best.

In Section 2, it was already explained that the NLP looks for specific concepts in the whole word graph, such as departure station, arrival station etc. Since these concepts are words, WG-WER would seem to be the most relevant evaluation measure. However, it is not necessary that the NLP recognizes every single word. Recognition of the above-mentioned crucial concepts will suffice. Although WG-WER is probably a better measure of the CSR performance than the other three indices mentioned previously, it is obvious that it is not an optimal measure. Indeed, the optimal measure would be a concept error rate for the word graph. In order to provide complete information about the performance of the CSR, the remaining three measures are also presented. The BS error rates give an idea of the quality of the phoneme models and bigrams, because the probabilities of the phonemes and bigrams are used to determine the BS from the WG. The SERs show how often the complete sentence is recognized correctly.

The different data bases were used to train language models (unigrams and bigrams) and the 38 acoustic models. Language models trained on data bases DBj will be called Lj. Phoneme models trained on data base DBn will be called Pn. In addition, phoneme models were trained on DB0 in combination with an application-specific data base DBm. These phoneme models will be called P0m.

The test data base was used to calculate error rates for various versions of the system (see Tables 2 and 3).

*Table 2.* Test-set perplexities and performance levels for the four different combinations of two phoneme models (P0 and P03) and two language models (L0 and L3).

| System | P0 + L0 | P03 + L0 | P0 + L3 | P03 + L3 |
|---|---|---|---|---|
| Perplexity | 317.90 | 317.90 | 65.84 | 65.84 |
| WG-WER | 26.16 | 23.78 | 13.56 | 13.32 |
| WG-SER | 42.20 | 37.80 | 25.00 | 24.60 |
| BS-WER | 49.04 | 41.28 | 27.18 | 23.66 |
| BS-SER | 66.00 | 55.60 | 41.80 | 37.00 |

*Table 3.* Test-set perplexities and performance levels for different phoneme models (Pi) and language models (Lj).

| System | P3 + L3 | P4 + L4 | P5 + L5 | P6 + L6 | P7 + L7 |
|---|---|---|---|---|---|
| Perplexity | 65.84 | 50.30 | 48.20 | 35.24 | 35.22 |
| WG-WER | 14.81 | 11.89 | 11.35 | 8.48 | 8.42 |
| WG-SER | 25.80 | 22.80 | 20.40 | 16.60 | 16.80 |
| BS-WER | 26.11 | 21.45 | 20.61 | 16.79 | 16.37 |
| BS-SER | 37.40 | 32.20 | 30.20 | 25.80 | 25.60 |

First, phoneme models (PMs) and language models (LMs) were trained on the Polyphone material (DB0) alone. The resulting error rates are given in column 'P0 + L0'. DB1 was not used to train PMs and LMs because the number of utterances in this data base is too small. Since the error rates for DB2 are very similar to those for DB3, they are not presented here. Although various tests were performed, here we will present only the results of the tests in which the test lexicon consisted of the words contained in the test data base. The reason for this is that these results best illustrate the gradual changes in the performance of the CSR.

We first wanted to test how important application-specific data are for training PMs and LMs. In order to do so we calculated the error rates for all four combinations of two sets of PMs (P0 and P03) and two sets of LMs (L0 and L3). The results are given in Table 2. DB0 was not used in training the second set of LMs (i.e., L3), because this data base contains no utterances that are relevant to the present application. In this case using DB0 in addition to DBn would only worsen the LMs. For PMs, on the other hand, it appeared that PMs trained on DBn and DB0 are better than those trained on DBn alone.

The test-set perplexities in Table 2 show that the LMs trained on application-specific data (i.e., L3) are better than those trained on DB0 (i.e., L0), as expected. Using DB3 in addition to DB0 to train the

PMs improves the performance (compare 'P0 + L0' with 'P03 + L0'). However, a much larger improvement is achieved if application-specific data are used to train the LM (compare 'P0 + L0' with 'P0 + L3'). An additional, albeit relatively small, gain in performance can be obtained when both the PMs and the LM are trained on application-specific data (compare 'P0 + L3' with 'P03 + L3'). On the basis of these results we may conclude that using application-specific data is more important for training the LM than for training the PMs.

Above we stated that PMs trained on DBn and DB0 are usually better than those trained on DBn alone. If we compare 'P03 + L3' of Table 2 with 'P3 + L3' of Table 3 we notice that this is indeed the case for DB3. However, as the size of the application-specific data base DBn increases, it becomes less important to use DB0 in addition to DBn to train the PMs. For this reason, only tests in which application-specific data were used are presented in Table 3. In Table 3 it can be observed that the test-set perplexity and the error rates gradually diminish as the size of DBn increases. However, for DB7 there is hardly any further improvement.

### *3.4. Evaluating the Performance of the Whole System*

In the previous section we have only dealt with the performance of a single component of the total system, i.e., the CSR module. In order to obtain a complete picture of the performance of the whole SDS different types of measures must be collected. For three measures it was specified a priori what the criteria were that the system had to meet.

First of all, it was specified that the system should not need a cold start (reboot) more often than once per two months, a criterion that is easy to measure by keeping a system log. Our system needed only two reboots in six months time. Second, the response time of the schedule data base had to be less than 3 seconds. A powerful workstation was needed to meet this requirement.

Third, the success rate of the system had to be greater than 80%. Success rate is the percentage of inquiries in which the callers get the information they asked for. In order to measure the success rate of the system a formal test was carried out with 500 subjects, all of whom were employees of KPN. They were selected so as to cover all regional language varieties. Subjects did not receive specific tasks to carry out. Instead they were simply requested to make up one or more realistic questions, and then to try and get the answer to these questions from the SDS. In total 647 calls were made during the test, in which 890 inquiries were presented. All calls were orthographically transcribed and evaluated.

The median duration of an inquiry was 140 seconds. This is slightly longer than the median duration of the calls in the present operator service. Unfortunately, we have no data about the relative time spent in collecting the input information and in providing schedule information. This lack of detail applies to both the operator and the automatic version of the service.

Of the 890 recorded inquiries 153 had to be left out of the evaluation because they were not considered to be serious attempts to obtain information (e.g., people playing with the system, inquiries in which multiple speakers talked simultaneously, and inquiries in which non-existing station names were used). For the remaining 737 inquiries 94.7% proved to be successful. This is far beyond the goal we had set of 80% successful queries.

## 4. Integration of the SDS in the Operational Service

Now that it has been shown that the SDS technology is capable of handling the majority of the calls relating to queries about schedules between train stations, inevitably the question arises whether and how the SDS should be integrated in the existing operator service. At the time of this writing a definitive decision about the introduction of the SDS is still pending. Yet, several considerations affecting the decision have been studied and discussed.

First of all, it is felt that the system can and should be improved before it can be made available commercially in a premium rate service. The performance of the CSR module leaves room for improvement, especially when it comes to recognizing the names of infrequently requested stations. Also, the noise robustness must be improved. Most probably, the error rates can be reduced by using cepstrum coefficients as the acoustic features, triphone models instead of monophones, and a more balanced bigram language model in combination with an improved pronunciation lexicon. Furthermore, we plan to record some 12,000 additional dialogs with the SDS to enlarge the training material. Apart from improving the current SDS, the functionality of the system must also be extended somewhat. At the very least the SDS should be able to handle callers asking for connections to large cities that do not have a train station. Finally, the way in which schedule information is presented to the caller can be improved, especially when the schedule involves more than one change of trains.

When the SDS goes on line it must be decided whether to advertise it as a new service, available under a new telephone number, or whether it should be accessed via the existing premium rate number. A new number has the advantage that callers will not be surprised when their call is answered by a machine whose functionality is limited to schedule information between train stations. Also, the tariff of this new number can be set independently of the cost of a call to the existing operator service. The single most important disadvantage of a new number is the cost incurred by the marketing and advertising actions needed to make that number and the functionality of the service known to the public. In the recent past, major efforts have been spent to inform the public about the existence of a single number for all transport information. Market surveys show that about 60% of the adult population knows the number.

Integration of the SDS under the existing access number causes problems as well. Since the SDS implements only part of the functionality of the operator service, all callers must be given the option to choose operator service or the SDS when they enter the queue. This is a major change in the present service. The welcome message must be changed so as to include a menu option, which is not exactly easy to explain (because it involves at least two messages: the alternative service is automatic, and its functionality is strictly limited). One might consider offering the choice only if the queue length exceeds a certain threshold. In that case the message does not need to be very short, because the caller will have to wait in the queue for at least a minute or so anyhow. Unfortunately, the network infrastructure of the premium rate network and the way in which the automatic call distributor is connected to the central office switch makes it very difficult to allow speech input for making the selection; thus, only callers who have a dual-tone multi-frequency (DTMF) phone will be able to select the SDS. To complicate the marketing issue even further, the present network infrastructure does not allow a change in the tariff after the selection has been made. Callers cannot be offered a cheaper rate if they opt for the SDS until the middle of 1998.

Irrespective of the way in which the SDS will be positioned, it should always allow for operator fall back if the machine fails to handle a call properly. For instance, the current stand alone prototype described above will give up if it records two consecutive failures to interpret the answer to the same question. In such a case the caller should be switched to an operator queue. It remains to be decided whether there can be a special operator queue for callers who failed to get through with the SDS. If such a priority queue is formed, it may be necessary to raise the number of failures on the first question, to prevent clever callers from using the SDS to jump the regular queue. It also remains to be determined whether callers should be given the option to go to an operator after having obtained the information on one or more train schedules.

## 5. Discussion and Conclusions

In this paper we have described the development of a system that can be used for automating part of an existing telephone-based service. An important characteristic of this system is that it was derived from a prototype that had originally been developed for German. Moreover, a bootstrapping approach for collecting application-specific material was adopted, instead of the usual Wizard-of-Oz scenario.

This bootstrapping method appears to have considerable advantages. First of all, no time is spent on building, testing, debugging, and implementing the Wizard-of-Oz simulation, because the real system is immediately realized. Second, the fact that the system used to collect the data is the real system has some important consequences. For example, the specifications of the data-collection system and the SDS are almost the same, which means that the properties of the signals collected for development (like e.g., background noise, signal-to-noise ratio) closely resemble those of the signals the final system will eventually have to handle. Furthermore, the whole application with all the modules is used from the beginning, and not just one or some of its components. In this way specific problems pop up at an early stage and can be solved before the final implementation takes place. Recognition problems related to variations in speech style are a case in point. For instance, the use of hyperarticulations or spelling pronunciations is unlikely to cause difficulties with a wizard—(s)he will understand the utterance anyway—but they can be problematic when the system is completely automatic (see also Shriberg et al., 1992). In other words, since many of these practical problems are specific for the implementation of the SDS, they are not likely to emerge when a Wizard-of-Oz simulation is used, which means that not all findings and experiences obtained with a bootstrap version can be obtained with a Wizard-of-Oz scenario. Finally, it is possible to collect speech material and to test, debug, and evaluate the system at the same time.

However, one important disadvantage of this approach is that it requires that the first version of the system work well enough to be used for data collection. We succeeded in making a suitable bootstrap version for the following reasons. First of all, we could use the German prototype as a starting point. Secondly, we could use our knowledge of this specific application and of the two languages involved, German and Dutch. This knowledge, together with the fact that German and Dutch are not very different, made it possible to localize a substantial part of the German prototype for Dutch. Thirdly, Dutch speech data bases, albeit not application-specific, were available. These appeared to be very useful to train the initial phoneme models. Finally, we used the data collected with the keyboard version. These data, together with our knowledge of Dutch and of this application, were employed to develop the initial bigram and the NLP module. It is possible that under less advantageous circumstances, this approach would be less successful than it turned out to be in our case.

On the basis of our experience, we can therefore conclude that the bootstrapping approach was very successful. Furthermore, we found that phoneme models trained with data which are not specific for the given application still perform reasonably well. However, this is not the case for the language models. A large gain in performance was obtained when the language models were trained with application-specific data. We also showed that the small test data base used in our research succeeded in revealing the relative improvements obtained with various versions of the system.

In conclusion, we are satisfied with the results of the tests so far. Our goal was to automate part of an existing service. In order to reduce the complexity of the task, we limited the domain to information about journeys from one train station to another. So far, it seems that it should be possible to automate this part of the service. However, we are still improving the system and the final field tests still have to be performed. In the near future we hope to be able to report positive results on the final evaluation of the system.

## Acknowledgments

Part of the research was carried out in the framework of two projects: the MLAP project MAIS and the Dutch Priority programme 'Language and Speech Technology', which is funded by the Netherlands Organization for Scientific Research (NWO). We would like to thank KPN research for recording and transcribing the dialogs. Furthermore, we would like to thank an anonymous reviewer, whose thorough review helped to improve the quality of this paper.

## References

Aust, H., Oerder, M., Seide, F., and Steinbiss, V. (1994). Experience with the Philips automatic train timetable information system. *Proceedings IVTTA'94 2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications*, Kyoto, Japan, pp. 67–72.

Aust, H., Oerder, M., Seide, F., and Steinbiss, V. (1995). A spoken language inquiry system for automatic train timetable information. *Philips Journal of Research*, *49*(4):399–418.

Baayen, R.H., Piepenbrock, R., and van Rijn, H. (1993). The CELEX lexical database (on CD-ROM). Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania.

Damhuis, M., Boogaart, T., Veld, C., Versteijlen, M., Schelvis, W., Bos, L., and Boves, L. (1994). Creation and analysis of the Dutch Polyphone corpus. *Proceedings International Conference on Spoken Language Processing (ICSLP)'94*, Yokohama, Japan, pp. 1803–1806.

den Os, E.A., Boogaart, T.I., Boves, L., and Klabbers, E. (1995). The Dutch polyphone corpus. *ESCA 4th European Conference on Speech Communication and Technology: EUROSPEECH'95*, Madrid, Spain, pp. 825–828.

Kerkhoff, J., Wester, J., and Boves, L. (1984). A compiler for implementing the linguistic phase of a text-to-speech conversion system. In H. Bennis and W.U.S. van Lessen Kloeke (Eds.), *Linguistics in the Netherlands*, pp. 111–117.

Konst, E.M. and Boves, L. (1994). Automatic grapheme-to-phoneme conversion of Dutch names. *Proceedings International Conference on Spoken Language Processing (ICSLP)'94*, Yokohama, Japan, pp. 735–738.

Ney, H. and Aubert, X. (1994). A word graph algorithm for large vocabulary, continuous speech recognition. *Proceedings International Conference on Spoken Language Processing (ICSLP)'94*, Yokohama, Japan, pp. 1355–1358.

Oerder, M. and Ney, H. (1993). Word graphs: An efficient interface between continuous-speech recognition and language understanding. *Proceedings ICASSP'93*, Minneapolis, USA, pp. 119–122.

Oerder, M. and Aust, H. (1994). A real-time prototype of an automatic inquiry system. *Proceedings International Conference on Spoken Language Processing (ICSLP)'94*, Yokohama, Japan, pp. 703–706.

Shriberg, E., Wade, E., and Price, P. (1992). Human-machine problem solving using spoken language systems (SLS): Factors affecting performance and user satisfaction. *Proc. Speech and Natural Language Workshop*, New York, USA: Harriman, pp. 49–54.

Steinbiss, V., Ney, H., Haeb-Umbach, R., Tran, B., Essen, U., Kneser, R., Oerder, M., Meier, H., Aubert, X., Dugast, C., and Geller, D. (1993). The Philips research system for large-vocabulary continuous-speech recognition. *ESCA 3rd European Conference on Speech Communication and Technology: EUROSPEECH'93*, Berlin, Germany, pp. 2125–2128.

Steinbiss, V., Tran, B., and Ney, H. (1994). Improvements in beam search. *Proceedings International Conference on Spoken Language Processing (ICSLP)'94*, Yokohama, Japan, pp. 2143–2146.

Steinbiss, V., Ney, H., Aubert, X., Besling, S., Dugast, C., Essen, U., Geller, D., Haeb-Umbach, R., Kneser, R., Meier, H.-G., Oerder, M., and Tran, B.-H. (1995). The Philips research system for continuous-speech recognition. *Philips Journal of Research*, *49*(4):317–352.

Strik, H., Russel, A., van den Heuvel, H., Cucchiarini, C., and Boves, L. (1996). A spoken dialog system for public transport information. *Proc. of the Dept. of Language and Speech*, Nijmegen, Vol. 19, pp. 129–142.