A SPOKEN DIALOGUE SYSTEM FOR PUBLIC TRANSPORT INFORMATION

Helmer Strik, Albert Russel, Henk van den Heuvel, Catia Cucchiarini and Lou Boves

In: H. Strik, N. Oostdijk, C. Cucchiarini, & P.A. Coppen (eds.)
Proceedings of the Department of Language and Speech
Vol. 19, pp. 129-142, Nijmegen, The Netherlands, 1996

A SPOKEN DIALOGUE SYSTEM
FOR PUBLIC TRANSPORT INFORMATION

Helmer Strik, Albert Russel, Henk van den Heuvel, Catia Cucchiarini and Lou Boves

Abstract

In 1995 our department was involved in two projects in the field of continuous speech recognition. The main aim of these two strongly related projects was the development of basic technology that can be used to build advanced telephone-based systems for providing information about public transport. A short description of the work carried out within these projects is provided in the present article.

1. Introduction

During the last decade the performance of spoken dialogue systems has improved substantially. At the moment, the quality of these systems seems to be able to support a number of simple practical tasks in small and clearly delimited domains. As a result, much effort is spent nowadays to develop prototype telephone-based information systems in different countries. These systems are reminiscent of the well-known Air Travel Information System (ATIS) task that has been a focal point in the American ARPA-project. In Europe two MLAP (Multi-Lingual Action Plan) projects concerning public railway information have been carried out, viz. RAILTEL and MAIS. These projects differ from the ATIS task in that they aim to construct truly interactive systems, accessible over the telephone.

There are many reasons why information about public transport is a suitable domain for testing spoken dialogue systems, of which only some are mentioned here. First of all, the domain can be limited in ways that are obvious for a caller, which is a necessary requirement to reach a sufficient performance level. In the Dutch system that we are developing, the domain is limited by restricting the information to travel between train stations. Furthermore, there is a large demand for information about public transport. For instance, in the Netherlands there is one nationwide telephone number for information about public transport. This number receives about 12 million calls a year, of which only about 9 million are actually answered. At the moment, all calls are handled by human operators. A substantial cost saving would be achieved if part of these calls could be handled automatically. Moreover, automatic handling would probably reduce the number of unsuccessful calls.

In the Netherlands ‘public transport information’ is virtually identical with ‘multi-modal from address-to-address information’. The human operators who provide the service must access a large database that contains the schedule information of all public transport companies in the country. Especially the fine-meshed local transport networks pose substantial problems, e.g. when specific bus stops must be identified. The complex dialogues that may be required to disambiguate destinations on the address level are far beyond what can be achieved with existing speech recognition, natural language processing, and dialogue management technology. Therefore, we have limited the domain of our experimental public transport information system to information about travels between train stations. However, we intend to enlarge that domain gradually, e.g. by adding metro stations in Amsterdam and Rotterdam, and by adding tram stops or the major inter-regional buses (Interliners).

2. General description of the system

The starting point of our research was a prototype developed by Philips Research Labs (Aachen, Germany). This automatic inquiry system can give information about the schedules of the German railways. Here we will only give a short description of the system. Further details can be found in Oerder and Ney (1993), Steinbiss *et al.* (1993), Aust *et al.* (1994), Ney and Aubert (1994), Oerder and Aust (1994), Aust *et al.* (1995), Steinbiss *et al.* (1995). Conceptually, the Spoken Dialogue System (SDS) consists of four parts (in addition to the telephone interface):

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

In the CSR module acoustic models (HMM's), language models (N-grams), and a lexicon are used for recognition. In the current version monophones are modelled by continuous density HMM's. However, it is also possible to use diphones or triphones as basic units. In the future we will investigate whether the performance of the system can be improved significantly by using context-sensitive models.

The lexicon contains orthographic and phonemic transcriptions of each word. Currently, there is exactly one phonemic transcription for each word. Using only one pronunciation variant is not optimal, since words are often pronounced in different ways. Therefore, we are now investigating how pronunciation variation can best be handled within the framework of this SDS (see Section 5).

The output of the CSR module, and thus the input to the NLP module, is a word graph. 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 it, 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.

The DM takes care of gathering all the necessary information to perform a database query. It does so 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 station to which station do you want 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 will then ask “At what time do you want to travel from Nijmegen to Amsterdam tomorrow?”. It thereby asks for the information it is still missing to do a database query. At the same time it gives the caller feedback about what the NLP did understand. If the NLP made a mistake, the caller can correct the system, e.g. by saying: “No, I want to travel the day after tomorrow”. If the caller does not correct the system, the DM decides that the NLP did understand the concepts departure_station, arrival_station, and date correctly. These concepts are then frozen. This means that the caller can no longer change their values. If the DM has all the information it needs, it will do the database query and report to the caller what connection(s) it found.

The information found in the database (and all other feedback 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. The two projects

The lion's share of the work described below has been carried out within the framework of two projects. A short description of these projects is given in this section. Because it is difficult to say exactly which part of the work is done in which project, we will only give a global description of the work carried out in each project.

3.1 MAIS

The European MLAP project MAIS (Multilingual Automatic Inquiry Systems) started on December 1st, 1994, and ended on December 1st, 1995. The MAIS consortium consisted of one technology provider: Philips Research Labs (Aachen, Germany); two public transport companies: SNCF (French railways) and NS (Dutch railways, later associated to the Dutch public transport information service, OVR); and three universities: RWTH (Aachen, Germany), IRIT (Toulouse, France), and KUN (Nijmegen, the Netherlands). The goals of the MAIS project were:

[1] to specify the requirements for an automated multilingual public transport information system that can be accessed over the telephone by the general public;

[2] to specify assessment procedures which can be used to measure users' satisfaction with the service; and

[3] to provide Dutch and French versions of the CSR, NLP, and DM modules, which could eventually be used to build laboratory demonstrators of train timetable information systems for these languages.

For aims [1] and [2] MAIS worked in close collaboration with the MLAP project RAILTEL. Starting point for [3] was a prototype developed by Philips Research Labs (Aachen, Germany), which already existed at the beginning of the project (see section 2). This prototype could provide information about the schedules of the German railways. The work described in Section 4.1 mainly took place within the framework of this project.

As a follow-up of the MAIS project (and partly also under the PP-TST project described below) we have worked on the improvement of the CSR and NLP modules, to a level on which they can be used to implement an operational laboratory system for Dutch. Such a system was needed in order to be able to collect task specific speech that can be used to bring the modules to a performance level that might be sufficient for actual deployment. A ‘training database collection system’ has been available in the Dutch Public Switched Telephone Network since December 1995.

3.2 PP-TST

The NWO Priority Programme ‘Language and Speech Technology’ (in Dutch: ‘Prioriteits-Programma Taal- en Spraak-Technologie’, PP-TST) is a five-year project which started in January 1995. The partners involved in this project are the Netherlands Organization of Scientific Research (NWO, Den Haag), Philips Corporate Research (PCR, Eindhoven), Royal Dutch PTT (KPN Research, Leidschendam), Nijmegen University (KUN, Nijmegen), Institute for Perception Research (IPO, Eindhoven), Groningen University (RUG, Groningen), and University of Amsterdam (UvA, Amsterdam).

The PP-TST aims at the development of advanced telephone-based information systems. One prominent feature of this programme is its attempt to achieve scientific as well as practical goals at the same time. The practical goal is to build a demonstrator of an interactive spoken language information system that can give travel information about public transport in the Netherlands. A number of increasingly powerful demonstrators are planned. From a scientific point of view, original contributions are envisaged in robust speech recognition over the telephone, natural language processing, and dialogue management in information-seeking dialogues. In the area of speech recognition, the focus will be on signal processing techniques to remove channel characteristics, on the one hand, and on explicit modelling of pronunciation variation, on the other. As for NLP aspects of the system, three approaches will be compared, viz. the AI-type approach presently implemented in the system, corpus-based parsing, and parsing using a conventional wide-coverage grammar. On the level of dialogue control it will be investigated how the communication with the user can be made more effective and user-friendly.

4. Building a Dutch SDS

In order to build and train an SDS for a certain application, a considerable amount of data is needed. For collecting these data Wizard-of-Oz scenarios are often used. However, within the framework of the current projects a different approach was chosen, which consists of the following five stages:

[1] make a first version of the SDS with available data
(which need not be application-specific)
[2] ask a limited group of people to use this system, and store the dialogues
[3] use the recorded data (which are application-specific) 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.

4.1 The first version of the SDS

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

4.1.1 CSR

The CSR component of the first version of the SDS was trained with 2500 utterances of the Polyphone database (Damhuis et al., 1994; den Os et al., 1995). The whole database is recorded over the telephone and consists of read speech and (semi-)spontaneous speech. For each speaker 50 items are available. Five of the 50 items were used, namely the so called phonetically rich sentences. Each subject read a different set of five sentences, selected so as to elicit all phonemes of Dutch at least once. The more frequent phonemes are produced much more often, of course. The speech recognizer used in this version of the system is a monophone mixture density HMM machine. As a first approximation, we trained about 50 acoustic models; they represent the phonemes of Dutch, plus two allophones of /l/ and /r/.

Note that the first version of the CSR is trained with read speech (and not spontaneous speech, as in the application) and that only very few sentences were related to the public transport domain. In the intended application the speech will be spontaneous and related to public transport information. Therefore, the data used to train the first version of the CSR cannot be considered application-specific.

Phonemic forms in the lexicon were taken from three different sources: (1) the names of stations from the ONOMASTICA database (Konst and Boves, 1994), (2) the lemma forms of other words from the CELEX database (Baayen *et al.*, 1993), and (3) for words that were not found in those two databases the phonemic forms were generated by means of our grapheme-to-phoneme converter (Kerkhoff *et al.*, 1984). Up to now, training and testing have been done completely automaticly, i.e., no attempts have been made to improve recognition rates by making the phonemic representations in the lexicon more homogeneous, nor by investigating the optimal set of monophone models. Furthermore, as was already noted above, there is only one phonemic transcription for each word, i.e., pronunciation variation is not modelled. Therefore, recognition scores obtained so far must be considered as rough indications of what can be obtained in an initial development job.

4.1.2 NLP

Since German and Dutch are quite 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 much more complicated. To illustrate this point some examples are mentioned here. For instance, each language has it own expressions for special days. In Dutch we have “koninginnedag” (birthday of the queen), “sinterklaas” (a festivity on December 5th or 6th), and “oudjaarsdag” (December 31th, literally: ‘old year day’). It is very common to say e.g. “de dag na koninginnedag” (literally: ‘the day after queen’s day’). Thus, the system had to be taught to recognize these expressions, which do not occur in German.

Furthermore, in different countries people assign a different meaning to time expressions like morning, afternoon, evening, and night. Because these concepts are used very often to indicate approximate time of departure or arrival, they should be defined and handled accordingly. For instance, in the German system ‘morning’ is interpreted as a time between 00:00 and 10:00, while in the Dutch system it is interpreted as a time from 04:00 to 12:00. The time between 00:00 and 04:00 is usually referred to as ‘night’ in Dutch.

Apart from different notions of time expressions, there are also differences in the way the German and Dutch databases are constructed. These two kind of differences interact, and lead to the following problem (which we call the time-frame problem). In order to construct a database query, the NLP must determine the date on which the caller wants to travel. It uses the system clock and the caller’s information to do so. The system clock is the internal clock of the computer on which the NLP software runs. It can provide the time and the calender date. Let us call the date provided by the internal clock D. The German system uses a database in which a day starts at 00:00 and ends at 23:59. These times fully coincide with the beginning and the end of a calendar day. However, in the Dutch system a database is used for which the day starts at 04:00 and ends at 03:59. This gives some tricky problems when you want to interpret time-related expressions from the caller. For instance, if the system asks a Dutch caller “when do you want to travel?”, and the caller answers “tomorrow”, the interpretation of tomorrow depends on the time at which the answer is given. If the caller says “tomorrow” between 04:00 and 23:59 (s)he really means tomorrow, i.e. the system should interpret this as D+1. However, if (s)he says “tomorrow” between 00:00 and 03:59, a Dutch caller usually means today and not tomorrow. Consequently, the system should not interpret this ‘tomorrow’ as D+1, but instead as D. The special status of the time frame 00:00 - 04:00 in the Dutch system made it necessary to review all the interpretations of time and date expressions in the original German system.

We were convinced that we could never figure out all the expressions Dutch people could use in order to get information about public transport just by introspection. At the same time, we did not have a large database available that could be used to look for all possible expressions. Therefore, it was decided to proceed as follows: A preliminary version of the grammar was made by translating some parts from German and by changing some other parts. This part of the SDS was then tested independently of the speech interface with a keyboard version of the dialogue system. People could log in on a system, type their questions on a keyboard, and get the replies from the system on the screen. Because people are likely to formulate their sentences differently when they speak or type, they were instructed to try to formulate the sentences as they would do if they had to pronounce them.

In this way we were able to test the grammar and to gather some text material that could be used to train the language model. 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. A nice example is that in the original German grammar there are 18 ways to give an affirmative answer and 7 ways to give a negative answer. Based on the log-files we have defined 34 affirmative answers and 18 negative answers for Dutch.

4.1.3 DM

For the bootstrap version of the system the German DM was translated literally into Dutch. Some adaptations appeared to be necessary, though. For instance, the interface to the public transport database had to be modified. Furthermore, some changes were required in the feedback to the caller. By way of illustration, in the German system train numbers are mentioned 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 feedback in the Dutch system.

As mentioned above, a database query is initiated only after all necessary information is available. Before an information item is considered as known and frozen, the caller is given explicit or implicit feedback about what the system thinks it has recognized. The caller can then disconfirm erroneous items and replace them with correct information.

4.1.4 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. An inventory was made of the phrases that together form the questions and replies the system should be able to produce. Recordings were made of these utterances spoken by a female speaker. In the SDS these recorded utterances are concatenated to generate the speech output of the system.

4.2 Improving the SDS

The first version of the SDS was put in the PSTN in December 1995. This version was trained with DB0, i.e. the 2500 Polyphone utterances. A small group of people received the telephone number of this system, and were requested to call it regularly. Their dialogues were recorded. In this way the databases DB1 to DB5 in Table 1 were collected. These databases are built up incrementally, which means that DB2 is a superset of DB1, DB3 of DB2, etc.

Table 1. Databases used during development of the SDS

Database	utterances	source	duration (hours:min)
DB0	2500	Polyphone	4:42
DB1	1301	application	0:41
DB2	5496	application	3:47
DB3	6401	application	4:35
DB4	8000	application	5:55
DB5	10003	application	7:20

For every utterance in the databases an orthographic transcription was made manually. Out-of-vocabulary words were detected automatically from the transcriptions. In this way words containing typing errors were found as well. All these typing errors were corrected. The out-of-vocabulary words were phonematized and added to the training lexicon, in order

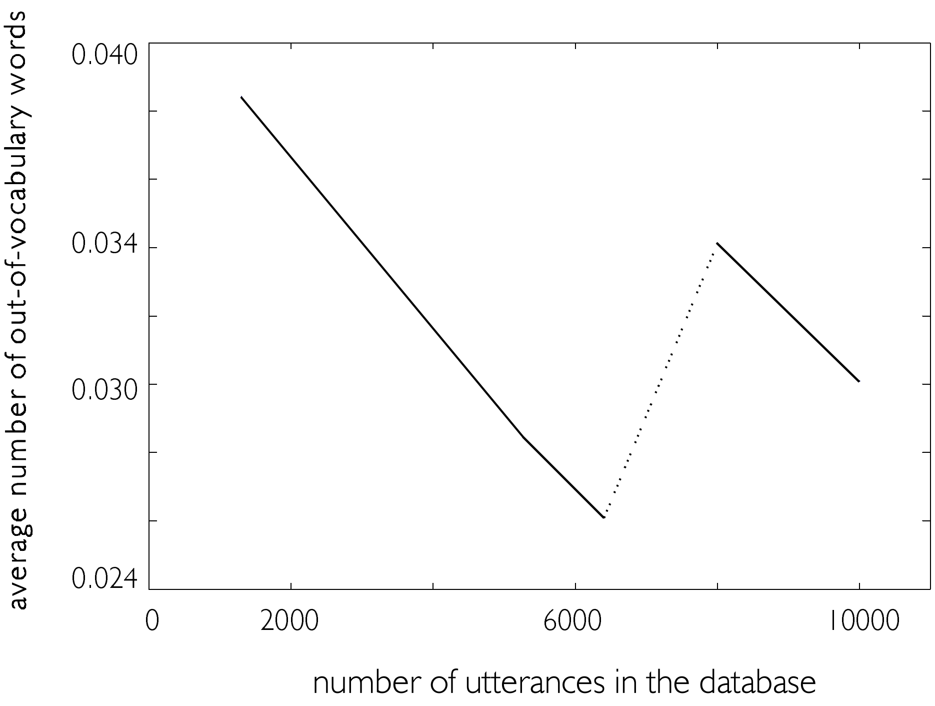

Figure 1. The average number of out-of-vocabulary words as a function of the number of utterances in the database.

to make it possible to use all the collected data for training the system. However, not all new words were added to the recognition lexicon. Only the words that were related to crucial concepts of the application were included in the recognition lexicon.

The average number of out-of-vocabulary words is shown in Figure 1. On the horizontal axis the number of utterances in the database is given (see Table 1). The vertical axis is the number of out-of-vocabulary words divided by the number of utterances, i.e. the average number of out-of-vocabulary words per utterance. It can be observed that the average number of out-of-vocabulary words is small. Apparently, we succeeded in making a bootstrap lexicon that contains most of the words used.

In Figure 1 one can also see that the average number of out-of-vocabulary words decreases as the number of utterances increases from 1301 to 6401. In the beginning a fair number of out-of-vocabulary 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 decreases gradually. 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 average number of out-of-vocabulary words turns out to increase first and to decrease again later on (see Figure 1).

Whenever a sufficient amount of new data was collected, language models 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 were replaced by the better ones.

In the early versions of the system we detected some syntactic constructions that where sometimes used by the callers but 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 of the DM module will be addressed in the near future.

4.3 Evaluating the performance of the CSR module

Part of the data collected with the on-line SDS was kept apart as a test database (500 utterances). The first test database was created by randomly selecting 500 utterances. The first evaluations were done with this test database. However, after some time we found out that this database was not well balanced, i.e. it contained a lot of utterances of a few speakers who often used the system in the beginning. That is why we decided to create a second (more balanced) test database, also containing 500 utterances. This database was used for later evaluations. The total number of words and characters (i.e. phonemes) in each database is approximately 1.700 and 10.000, respectively. The number of different words in test database 1 and 2 is 298 and 299, respectively. This means that test databases 1 and 2 are equally large.

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.

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.

Note that these results were obtained with a version of the system which was available at the beginning of the MAIS and PP-TST projects. Thanks to the use of new features, the performance of the CSR module has now improved. However, this improved version has not been used for the research described in the present article. Still, the research findings reported here apply to the improved version too, because they concern basic aspects of the system.

The different databases were used to train language models and phoneme models. In all cases the inventory of phonemes remained the same. Language models trained on databases DBj will be called Lj. Phoneme models trained on database DBn will be called Pn. In addition, phoneme models were trained on DB0 in combination with an application-specific database DBm. These phoneme models will be called P0m.

With test database 1 the error rates for several versions of the system were obtained (see Table 2). First, the phoneme and language models were trained with the Polyphone material (DB0). The resulting error rates are given in column 2. DB1 was not used to train phoneme models and a language model because the number of utterances in DB1 (i.e. 1301) was too small.

Table 2. Performance level for different phoneme models (Pi) and language models (Lj). Evaluation is done with test database 1.

System	P0 + L0	P02 + L0	P02 + L2	P03 + L2	P03 + L3	P3 + L2	P3 + L3
WG - WER	20.59	18.36	6.72	6.94	6.94	6.94	6.94
WG - SER	40.00	36.60	16.00	15.20	15.60	16.20	15.40
BS - WER	39.87	31.45	14.73	15.43	15.70	16.41	14.84
BS - SER	65.00	54.20	28.00	29.00	28.60	26.00	26.40

Training the phoneme models on both the Polyphone data (DB0) and application-specific data (DB2) reduces the error rates (compare column 3 to column 2). However, a much larger reduction in the error rates is obtained by training the language model on DB2 (compare column 4 with 2 and 3). The conclusion is that application-specific data is much more important for training the language models than for training the phoneme models. Other comparisons of performance levels with different databases confirmed this conclusion.

Increasing the number of utterances in the database from 5496 to 6401 does not have much effect on the level of performance (compare columns 5 and 6 with column 4). This could be due to the fact that the amount of added utterances (i.e. 905 utterances) is small compared to the size of the database. What is more important is that performance does not deteriorate if the Polyphone material is left out when training the phoneme models (compare columns 7 and 8 with columns 5 and 6, respectively). On the contrary, BS-WER is even slightly better for phoneme models trained with DB3 (given in columns 7 and 8), compared to phoneme models trained with DB3 and DB0 together. Therefore, we decided not to use the data from the Polyphone database anymore for the current application.

Table 3. Performance levels for different phoneme models (Pi) and language models (Lj). Evaluation is done with test database 1 (column 2: old) and 2 (columns 3-5: new).

testDB	old	new		
System	P3 + L3	P3 + L3	P4 + L4	P5 + L5
WG - WER	6.94	8.87	6.81	6.69
WG - SER	15.40	17.80	14.40	13.80
BS - WER	14.84	15.27	12.93	14.02
BS - SER	26.40	25.40	24.20	24.60

At this moment test database 1 was replaced with test database 2. For phoneme models P3 and language model L3 the error rates obtained with test database 2 were higher than those obtained with test database 1, except for BS-SER (see Table 3, compare columns 2 and 3). However, increasing the size of the training database to 8.000 utterances led to a better performance. The effect of increasing the database to 10.003 utterances was small. Evaluation results for a third test database, and for a larger training database (consisting of 21.288 utterances) are presented in Strik *et al.* (1996).

5. Pronunciation variation and non-speech sounds

Apart from the work described in the previous section, some other research was carried out in order to improve the SDS. In the present section we will only give a short description of some issues related to modelling pronunciation variation and recognizing non-speech sounds.

In order to obtain the phonemic representations of the words in the lexicon, we first checked whether these words were present in two existing databases, namely CELEX (Baayen *et al.*, 1993) and ONOMASTICA (Konst and Boves, 1994). Phonemic transcriptions of the words that could not be found in these two databases were derived by using the grapheme-to-phoneme conversion rules developed at our department (Kerkhoff *et al.*, 1984). The output of the rules was then checked and, if necessary, corrected by hand. There are several reasons why a lexicon obtained in this way is not optimal for speech recognition:

1. since the phonemic transcriptions are obtained from different sources, they are likely to be inconsistent;
2. for each entry in the lexicon only one pronunciation variant is stored, while in practice people will pronounce words in many different ways;
3. the pronunciation variant present in the lexicon is not always the optimal one for speech recognition (see e.g. Cohen, 1989).

For instance, the policy adhered to in the ONOMASTICA project was to limit reduction phenomena to the bare minimum. As a result many of the station names are represented by overly formal phonemic forms. By way of illustration, we will give some examples of recognition errors which are most probably due to pronunciation variation (in our system and in the examples below SAMPA is used as the computer phonetic alphabet).

In one dialogue a person did not succeed in convincing the SDS that he wanted to go to a place called Geldrop. Although he tried several times, the system did not manage to recognize the word, because the speaker in question did not say [GELdrOp] (the transcription of Geldrop in the lexicon), but [GELd@rOp]. Although this is only a minor difference for human listeners, who are expert speech recognizers, this example illustrates that a small difference in pronunciation (i.e. insertion of a schwa) can have serious consequences for automatic SDS (i.e. recognizing the wrong place name). Reduction processes, which are very common in spontaneous speech, also caused several problems. For instance, many people say something like [xujdAx] instead of [xud@ndAx] or [xuj@ndAx], which are more careful pronunciations of the Dutch word “goedendag” (a greeting which literally means “nice day”). Another example of severe reduction is the pronunciation of Amsterdam as [Ams@dAm] instead of [Amst@rdAm].

As spontaneous speech exhibits a considerable amount of pronunciation variation, the speech recognizer's performance can be improved if the variation is properly taken into account. For this reason part of our research on speech recognition is now concentrated on modelling pronunciation. A first step in this direction consists in making an inventory of possible pronunciation variants present in spontaneous speech. Although a large amount of pronunciation variation in Dutch is described in the literature (see, e.g., Booij, 1995), we also found variation forms which probably have not been described before (see Cucchiarini and Van den Heuvel, 1996).

Besides modelling pronunciation variation, correct recognition of non-speech sounds is also very important. We encountered many examples of this phenomenon, some of which are mentioned here. One sentence “ja dat klopt” (‘yes, this is correct’) was followed by a long interval of breath noise after the last word. The system recognized the words ‘dat’ en ‘klopt’ correctly. But the system also recognized the final bit of non-speech as speech, and thus recognized: “nee dat klopt niet” (‘no, this is not correct’). This is exactly the opposite of what was meant. Furthermore, non-speech sounds were found very often at the beginning of an utterance. In many cases a speaker starts an utterance by inhaling. This inhaling noise is often followed by a lip-smack. Some preliminary experiments revealed that modelling (and recognizing) these non-speech sounds does improve the performance of the SDS.

6. Discussion and conclusions

In this paper we have described the development of a system that can be used for automatizing 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, an alternative approach for collecting application-specific material was adopted, instead of the usual Wizard-of-Oz scenario.

This alternative method appears to have considerable advantages. First of all, no time is spent on building, testing, debugging, and implementing the WOZ simulation. Instead, the real system is immediately realized. Consequently, the system used to collect the data is the real system and not some imitation, the specifications of the data-collection system and the final SDS are the same, and thus 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 practical problems pop up at an early stage and can be solved before the final implementation takes place. Many of these practical problems are specific for the implementation of the SDS. Therefore, most of them will not turn up when a WOZ simulation is used. In short, not all findings and experiences obtained with a bootstrap version can be obtained with a WOZ simulation. 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 should work well enough to be used for data collection. We succeeded in making a suitable bootstrap for the following reasons. Firstly, because we could use the German prototype as a starting point. Secondly, because we had knowledge about German, Dutch, and this specific application. These three types of 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, because speech databases, albeit not application-specific, were available. They were used especially to train the phoneme models. Finally, because we used the data collected with the keyboard version. These data, and our knowledge of Dutch and this application, were used to develop the 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 bootstrap 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 databases used in our research succeeded in revealing the relative improvements obtained with various versions of the system. However, the absolute numbers for the performance levels differed between the two test databases. Therefore, it is probably better to use more than one database for testing.

Finally, we are satisfied with the results of the tests so far. Our goal was to automatize 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 automatize 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.

Acknowledgements

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).

References

Aust, H., M. Oerder, F. Seide and V. Steinbiss (1994), ‘Experience with the Philips Automatic Train Timetable Information System’, in: *Proceedings IVTTA'94 2nd IEEE* *workshop on interactive voice technology for telecommunications applications*, Kyoto, 67-72.

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

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

Booij, G. (1995), *The phonology of Dutch*. Oxford: Clarendon Press.

Cohen, M.H. (1989), *Phonological structures for speech recognition*, PhD dissertation, University of California, Berkeley.

Cucchiarini, C. and H. van den Heuvel (1996), ‘/r/-deletion in standard Dutch’, in: *Proceedings of the Department of Language and Speech*, Nijmegen University, 19, this issue.

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

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

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

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

Oerder, M. and H. Aust (1994), ‘A Real-time Prototype of an Automatic Inquiry System’, in: *Proceedings International Conference on Spoken Language Processing (ICSLP) '94*, Yokohama, 703-706.

Oerder, M. and H. Ney (1993), ‘Word graphs: an efficient interface between continuous-speech recognition and language understanding’, in: *Proceedings ICASSP'93*, Minneapolis, 119-122.

den Os, E.A., T.I. Boogaart, L. Boves and E. Klabbers (1995), ‘The Dutch Polyphone corpus’, in: ESCA *4th European Conference on Speech Communication and Technology: EUROSPEECH ‘95*, Madrid, 825-828.

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

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

Strik, H., A. Russel, H. van den Heuvel, C. Cucchiarini and L. Boves (1996), Localizing an automatic inquiry system for public transport information. To appear in: *Proceedings International Conference on Spoken Language Processing (ICSLP) '96*, Philadelphia.