

# LOCALIZING AN AUTOMATIC INQUIRY SYSTEM FOR PUBLIC TRANSPORT INFORMATION

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

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

## ABSTRACT

This paper reports on the development of a spoken dialogue system for providing information about public transport in the Netherlands. It is explained how a German prototype was adapted for Dutch. Emphasis is laid on the specific approach chosen to collect speech material that could be used to gradually improve the system. The pros and cons of this method are discussed.

## 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 and the major inter-regional buses (Interliners). The lion's share of the research has been done in the framework of two projects (which are briefly described in [1]): the European MLAP project MAIS and the Dutch Priority programme 'Language and Speech Technology'.

## 2. GENERAL DESCRIPTION OF THE SDS

The starting point of our research was a prototype developed by Philips Research Labs (Aachen, Germany), which can provide information about the schedules of the German railways (for further details concerning this system, see [2, 3, 4, 5, 6]). 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 all words. 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 [1]).

The output of the CSR module is a word graph. In the NLP module a stochastic attributed context-free grammar is used to parse this word graph. The main goal of the grammar is to find the information that is needed to perform the right query on the database. Therefore, it is not necessary that all words are recognized and understood correctly. It is sufficient that important concepts (like e.g. origin, destination, and time of departure or arrival) are recognized correctly.

The DM module checks whether all information needed to perform a query on the database is present (i.e. whether all slots are filled). If this is not the case, the system asks the caller explicitly for the missing information. When all slots are filled, the system accesses the database.

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. 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 project a different approach was chosen, which consists of the following five stages:

1. make a first version of the SDS with available data
2. ask a limited group of people to use this system and store the dialogues
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 description of the system developed by Philips Research 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.

**CSR.** The CSR component of the first version of the SDS was trained with 2500 utterances of the Polyphone database [7]. 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 to be application-specific.

Phonemic forms in the lexicon were taken from the ONOMASTICA database (station names) [8], from the CELEX database (lemma forms of the other words) [9] or generated by means of our grapheme-to-phoneme converter. Up to now, the training and testing have been completely automatically, 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.

**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 more complicated. To illustrate this point some examples are mentioned here. For instance, each language has its 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.

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. People could log in on a system, type their questions on a keyboard, and get the replies from the system on the screen (keyboard version). 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.

**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.

He can then disconfirm erroneous items and replace them with correct information.

**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.

### 3.2 Improving the SDS

The first version of the SDS was put in the PSTN (Public Switched Telephone Network) in December 1995. This version was trained with DB0, 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 then recorded. In this way the databases DB1 to DB6 in Table 1 were collected. These databases are built up incrementally, which means that DB2 is a superset of DB1, DB3 of DB2, etc.

| Databases | Utterances | Source      | Duration |
|-----------|------------|-------------|----------|
| DB0       | 2500       | Polyphone   | 282 min. |
| DB1       | 1301       | application | 41 min.  |
| DB2       | 5496       | application | 227 min. |
| DB3       | 6401       | application | 275 min. |
| DB4       | 8000       | application | 355 min. |
| DB5       | 10003      | application | 440 min. |
| DB6       | 21288      | application | 921 min. |

**Table 1:** Databases used during development of the SDS.

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 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.

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 some cases certain syntactic constructions were not handled correctly by the NLP. In these cases the NLP was adjusted in order to make it possible to recognize these expressions. In this way CSR and NLP were gradually improved.

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 database (500 utterances). These 500 utterances contain 296 different words. The total number of words and characters in this test database 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 database.

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.

Note that these results were obtained with a version of the system which was available at the beginning of the project. 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  $Db_j$  will be called  $L_j$ . Phoneme models trained on database  $Db_n$  will be called  $P_n$ . In addition, phoneme models were trained for DB0 in combination with an application-specific database  $DB_m$ . These phoneme models will be called  $P_{0m}$ .

The test database was used to calculate error rates for various versions of the system (see Table 2). First, phoneme models (PMs) and language models (LMs) were trained with the Polyphone material (DB0) only. 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 (1301) was too small. Subsequently, the application-specific databases DB2 and DB3 were used for training. For each of these databases four systems are compared ( $DB_n$  is either DB2 or DB3):

1.  $P_{0+L_0}$ : PMs and LM trained on DB0
2.  $P_{0n+L_0}$ : PMs trained on DB0 +  $DB_n$ , LM trained on DB0
3.  $P_{0+L_n}$ : PMs trained on DB0 and LM trained on  $DB_n$
4.  $P_{0n+L_n}$ : PMs trained on DB0 +  $DB_n$ , LM trained on  $DB_n$

In situations 3 and 4 DB0 was no longer used in training the LM, because this database contains almost no utterances that may be relevant to the present application. Using DB0 in addition to  $DB_n$  in this case would only worsen the LM. For PMs, on the other hand, it appears that the PMs trained on  $DB_n$  and DB0 are better than those trained on  $DB_n$  alone.

| System     | P0+L0  | P02+L0 | P0+L2 | P02+L2 | P03+L0 | P0+L3 | P03+L3 | P3+L3 | P4+L4 | P5+L5 | P6+L6 |
|------------|--------|--------|-------|--------|--------|-------|--------|-------|-------|-------|-------|
| perplexity | 317.90 | 317.90 | 65.80 | 65.80  | 317.90 | 65.84 | 65.84  | 65.84 | 50.30 | 48.20 | 35.24 |
| WG-WER     | 26.16  | 24.85  | 13.32 | 12.84  | 23.78  | 13.56 | 13.32  | 14.81 | 11.89 | 11.35 | 8.48  |
| WG-SER     | 42.20  | 39.80  | 25.20 | 23.80  | 37.80  | 25.00 | 24.60  | 25.80 | 22.80 | 20.40 | 16.60 |
| BS-WER     | 49.04  | 42.17  | 27.42 | 24.37  | 41.28  | 27.18 | 23.66  | 26.11 | 21.45 | 20.61 | 16.79 |
| BS-SER     | 66.00  | 54.20  | 42.20 | 37.40  | 55.60  | 41.80 | 37.00  | 37.40 | 32.20 | 30.20 | 25.80 |

**Table 2:** Test-set perplexities and performance levels for different phoneme models (Pi) and language models (Lj).

Using application-specific data for training the PMs alone slightly improves the level of performance (compare columns 3 and 6 with column 2). However, a much larger improvement is achieved if application-specific data are used to train the LM alone (compare column 4 with 3 and 7 with 6). Compared to the latter systems, the gain in performance is again small when also the PMs are trained on the application-specific data (compare column 5 with 4 and 8 with 7). Therefore, we may conclude that using application-specific data is more important for training the LM than the PMs.

Increasing the number of utterances in the database from 5496 to 6401 does not have much effect on the level of performance (compare columns 6, 7, and 8 with 3, 4, and 5, respectively). This could be due to the fact that the number of added utterances (905) is small compared to the size of the database. What is more important is that performance does not deteriorate much if the Polyphone material is left out when training the phoneme models (compare columns 8 and 9). Increasing the amount of training material gradually improves the level of performance (compare columns 10, 11, and 12 with column 9).

#### 4. DISCUSSION AND CONCLUSIONS

In this paper we have described the development of an automatic system for providing information about public transport in the Netherlands. Important characteristics of this system are that it was derived from a prototype that had originally been developed for German and that 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 the WOZ simulation. Instead, the real system is immediately realized. The whole application with all the modules is used from the beginning, and not just one component. In this way many practical problems pop up at an early stage, and can be solved before the final implementation takes place. Furthermore, the system used to collect the data is the real system and not a simple imitation. 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. Thirdly, because German and Dutch are very

similar. Fourthly, because speech databases, albeit not application-specific, were available. Finally, because we used the data collected with the keyboard version. It is possible that under less advantageous circumstances, this approach would be less successful than it turned out to be in our case.

#### 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).

#### 5. REFERENCES

1. Strik, H., Russel, A., van den Heuvel, H., Cucchiari, C., and Boves, L. "A spoken dialogue system for public transport information", *Proc. of the Dept. of Language and Speech*, Vol. 19, 129-142, 1996.
2. Oerder, M. and Ney, H. "Word graphs: an efficient interface between continuous-speech recognition and language understanding," *Proc. ICASSP'93*: 119-122, 1993.
3. Steinbiss, V., Ney, H., Haeb-Umbach, R., Tran, B., Essen, U., Kneser, R., Oerder, M., Meier, H., Aubert, X., Dugast, C., and Geller, D. "The Philips research system for large-vocabulary continuous-speech recognition", *Proc. EURO-SPEECH'93*: 2125-2128, 1993.
4. Aust, H., Oerder, M., Seide, F., and Steinbiss, V. "Experience with the Philips Automatic Train Timetable Information System", *Proc. IVTTA'94*: 67-72.
5. Ney, H. and Aubert, X. "A word graph algorithm for large vocabulary, continuous speech recognition", *Proc. ICSLP'94*: 1355-1358, 1994.
6. Oerder, M. and Aust, H. "A Real-time Prototype of an Automatic Inquiry System", *Proc. ICSLP'94*: 703-706, 1994.
7. den Os, E.A., Boogaart, T., Boves, L., and Klabbers, E. "The Dutch Polyphone corpus", *Proc. EUROSPEECH'95*, 825-828, 1995.
8. Konst, E.M. and Boves, L., "Automatic grapheme - to - phoneme conversion of Dutch names", *Proc. ICSLP'94*, 735-738, 1994.
9. Baayen, R.H., Piepenbrock, R., and van Rijn, H., *The CELEX lexical database (on CD-ROM)*, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, 1993.