

## A REALTIME PROTOTYPE OF AN AUTOMATIC INQUIRY SYSTEM

*Martin Oerder, Harald Aust*

Philips GmbH Forschungslaboratorien  
P.O. Box 1980, D-52021 Aachen, Germany \*

### ABSTRACT

We have developed an automatic system for train-timetable information over the telephone that provides accurate connections between 1200 German cities. The caller can talk to it in unrestricted, natural, and fluent speech, very much like he or she would communicate with a human operator, and is not given any instructions beforehand.

The speech recognizer of this system creates a word graph and passes it on to the speech understanding component. It is then parsed with a stochastic attributed context-free grammar that is used as a language model, to identify the relevant parts, and to compute their meaning. A dialogue manager analyses the results and either comes up with a new question or accesses the timetable database. The answer to this query, as well as the questions, are converted to spoken language by replaying appropriate pre-recorded phrases.

### 1. INTRODUCTION

Automatic inquiry systems are systems that people can call in order to obtain a certain information, without a service representative being involved. To this end, the system has to create a database query from the user's input and present the results to him or her.

While systems of this kind are already in operation, they are not very user-friendly. Callers have to interact with them either by pushing keys on their touch-tone telephone, or by uttering one of just a few words the system can understand. The ensuing dialogue is usually menu-driven, rigidly structured, and accompanied by lengthy explanations.

By contrast, the system described in this article allows users to talk in unrestricted, natural, and fluent speech, very much like they would converse with a human operator. Our prototype installation works in real-time and provides information on train connections between the 1200 most important German cities.

\*The work reported in this paper was in part supported by the German Ministry of Research and Technology (BMFT) under contract No. 01 IV 102 B.

### 2. SYSTEM ARCHITECTURE

Unlike more traditional areas of automatic speech recognition, like dictation or voice control, an inquiry system does not only need recognition, but also speech understanding, dialogue control, and speech output capabilities. Of particular interest, then, is the way of interaction of these basic components, since an appropriate architecture should be expected to help increase the system's overall performance.

This is especially important for the integration of the recognition and understanding parts. An automatic inquiry system must, of course, be speaker independent. The quality of the incoming speech signal is low because of the limited-bandwidth, and often noisy, telephone line. The vocabulary may comprise several thousand words, yet the system has to operate at least close to real-time. These unfavorable conditions result in a relatively poor recognition performance — with a word error rate of about 25% in our current system — that one can hope to improve by exploiting the knowledge of the understanding component for recognition purposes, effectively using it as a language model.

We do not, however, integrate the different components of our system directly. On the contrary, we separate them into individual modules that are executed sequentially (Fig. 1). Well-defined interfaces ensure that the system can be easily maintained and that entire modules can be exchanged without affecting the remaining parts. This way, we avoid complicated and error-prone interface protocols inherent in more complex architectures.

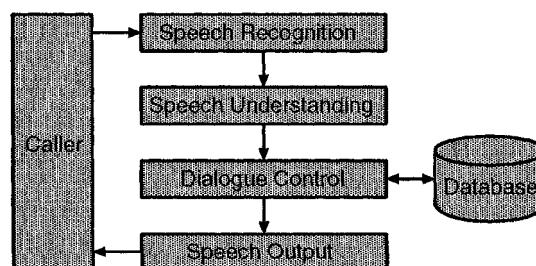


Figure 1: The system architecture.

### 3. SPEECH RECOGNITION

For speech recognition, we employ the PHICOS system of Philips Research. It uses Hidden Markov Models, continuous mixture densities, and a tree-organized beam search [9], and is based on the technology developed for Philips Dictation Systems' SP6000 dictation product [13].

This recognizer was modified in several respects for the inquiry system application. First of all, the feature extraction had to be adapted to the telephone-line bandwidth. We now use 14 spectral values, augmented by signal energy as well as first and second order time differences. Secondly, the system had to be trained for a speaker-independent operation over the telephone network. While the lack of appropriate data was a problem in the beginning, we have now some 10 hours of training material at our disposal.

Instead of simply computing the single best sentence, the recognition module generates *word graphs* [10] as its output. A word graph is a directed acyclic graph whose nodes represent points in time and whose edges are labelled with a word and an accompanying negative logarithm of an acoustic probability.

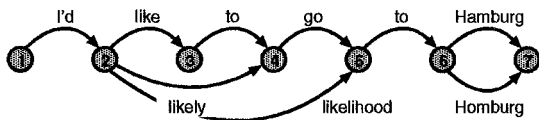


Figure 2: A sample word graph. The scores have been omitted for clarity.

These structures are created in a two-step process: a *word hypotheses generator* produces scored word hypotheses at a rate of several thousand per spoken word; a *word graph optimizer* then reduces the amount of hypotheses, so a relatively sparse word graph is obtained. This optimization does not need to access the acoustic models anymore, and it does not require major computational resources, either. For these reasons, our recognizer is comparable in overall speed to one which outputs just the best sentence.

The density of a word graph, measured in the average number of edges created for a spoken word, can be adjusted by tuning the relevant pruning parameters. In our prototype, graphs with about 10 edges per word turned out to be most effective.

### 4. SPEECH UNDERSTANDING

The word graph that was created by the speech recognition component is then passed on to the understanding module whose task it is to find the best path through the graph, and to determine its meaning.

This requires some kind of parsing. We do not, however, try to parse an entire sentence. Instead, we are only looking for those words or sequences of words that are usually used when someone expresses a certain aspect of a request for a timetable information. Examples

are “tomorrow” to indicate the date of travel or “to Hamburg” to specify the destination. These *concepts* may occur in arbitrary order and can be separated by *filler words* that are meaningless with respect to the creation of the database query, like “Good morning” or “I want to go”.

This technique has two major advantages: sentences that are grammatically incorrect — which is rather common since we are dealing with spontaneous speech —, or insufficiently recognized, can still, at least partly, be understood. And the method is computationally inexpensive: on average, our current system needs just 28 ms to understand a sentence.

#### 4.1. Stochastic Grammars for Concepts

We use a context-free phrase-structure grammar to specify the individual concepts. For each of them, a distinct start symbol is defined, so we really have a set of grammars that can, however, share subordinate rules. Note that the grammar is a semantic rather than a syntactic one since it does not describe the structure of a sentence but its meaning.

With this grammar, an incoming word graph is transformed into a *concept graph* whose nodes are identical to those of the word graph; its edges, though, are not labelled with words but with instances of concepts. They also have a score that is composed of two components, the first of which being the acoustic probability of the concept instance as computed by summing up the word graph scores of the associated words. The other part consists of a language model probability. To compute it, we extend our grammar to a *stochastic grammar* [6] where every rule has a (negative-logarithmic) likelihood that can be automatically trained [7]. This value indicates the probability that the rule is applied, given the left-hand-side non-terminal. By summing up the scores of all rules that were used for a particular derivation, a probability can be computed and added to the acoustic score of the concept graph arc.



Figure 3: Concept graph.

#### 4.2. Filler Arcs

In general, of course, there is no path from the first to the last node of the concept graph anymore, since concepts will not always be adjacent to each other. Besides, finding an instance of a certain concept does not necessarily mean that the corresponding words were really said: they may appear in the word graph solely because of recognition errors. Therefore, we must be able to bridge gaps in the graph, as well as bypass the concept edges.

To achieve this, we introduce *filler arcs*. These edges are labelled with an empty concept; their score is equal

to the acoustic score of the most probable path through the word graph between their start node and their end node.

For every concept in the graph, a parallel filler arc is inserted. Also, begin and end of the concept graph are connected with every concept, and the same is done for all of those concepts that were not interlinked previously.

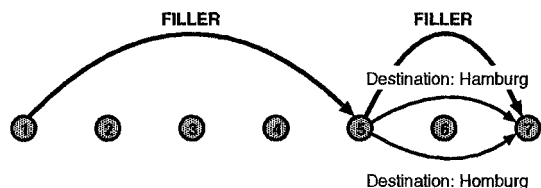


Figure 4: Completed concept graph.

### 4.3. Selection of the Best Path

The thus created *completed concept graph* now serves as the basis for finding the best path. If one simply took the path with the smallest sum of the scores of the contributing arcs, however, this would result in filler arcs only.

We avoid this by introducing a *filler penalty*: a time-proportional value is added to every filler edge. The time proportionality expresses that we expect words carrying a meaning with respect to our database query, and that sequences of insignificant words are increasingly unlikely the longer they become.

In addition to the acoustic values from the word graph and the language model scores from the stochastic grammar, both of which make up the concept arc scores, a *concept bigram score* contributes to the computation of the best path. This is done to acknowledge that certain concept sequences occur more frequently than others. Naturally, these probabilities can be trained automatically, too.

This simple method works surprisingly well, provided that the value of the filler penalty is chosen carefully.

### 4.4. Determining the Meaning

We have now found the most probable sequence of concepts. What we need in order to create the database query, however, is not the words the concepts consist of, but their meaning.

To facilitate their determination, we extend our grammar to an *attributed grammar* [6]: every non-terminal may have an arbitrary number of *attributes*, and every (syntactic) rule may have associated *semantic rules*. They allow the computation of attributes of the left-hand-side non-terminal out of already known attributes of the right-hand-side constituents which are accessed by writing down the appropriate non-terminal, followed by the attribute name (Fig. 5).

With such a grammar, an expression is parsed top-down, and its meaning is subsequently computed bottom-up.

```

<Time> ::= (2.56) <prep> <Hour> <Minute>
         time := <Hour>.number * 60 + <Minute>.number
<Hour> ::= (7.61) one | (6.38) two | ...
         number := 1 | 2 | ...
<Minute> ::= (19.31) fifteen | (17.22) twenty | ...
          number := 15 | 20 | ...
<prep> ::= (1.64) at | (2.11) after | ...

```

Figure 5: A simple stochastic attributed grammar for time expressions. The parenthesized values are negative logarithms of the likelihoods of the corresponding rules.

### 4.5. Representing Dates

While times can be expressed by integers (e.g. 2215 for 10:15 pm), and station names by character strings, things are more complicated for dates. Because of different lengths of months, as well as leap years, integers cannot readily be used, and there is no other evident data type that allows a straightforward modeling, either. Consequently, we defined our own.

A date consists of year, month, day, weekday, and duration. Each of these components may be undefined. New dates are then put together using the already known values of existing ones, and completed, if necessary, with the current date. With this approach, we can even model fairly complex phrases. “Next week”, for example, would be expressed as “the seven-day period beginning with the first monday after the current date”; “three days before christmas” as “the next 25th of december after the current date, minus three days”.

## 5. DIALOGUE CONTROL

Once we found the concepts and their meaning in the spoken sentence, it is theoretically possible to create the database query. There may, however, be several difficulties that prevent just this:

- Information needed for the query’s construction may be missing; either because the caller simply did not provide it, or because it was not retrieved during the recognition and understanding processes.
- For similar reasons, there may be ambiguous or contradictory data like two different destinations.
- Because of the recognizer’s relatively high error rate, it is important that the system can verify what it believes it understood. Otherwise, an incorrect query may result.

It is, of course, self-evident what must be done to overcome these problems: we need a system that can ask questions and involve the caller in a *dialogue*.

For reasons of user friendliness, it is highly desirable for this dialogue to be flexible, natural, and capable of adapting itself to the user’s utterances. Unfortunately, this may result in hundreds of different questions that can follow in almost arbitrary order, leading to tens of

thousands of possible dialogue situations that cannot be represented in any convenient way.

We have therefore developed a special programming language for dialogues in automatic inquiry systems: most aspects of this subclass of general dialogue are specified in a declarative way. The interpreter then takes care of the selection of appropriate questions, handles the results it gets from the speech understanding module, and finally creates the database query. With this description formalism, the complexity of a dialogue representation is greatly reduced.

## 6. SPEECH OUTPUT

Both the results from the database query and the questions generated by the dialogue manager are passed to the speech output component in the form of complete, written sentences. Any available speech synthesis program can then be used to convert these words into "spoken" language.

Real text-to-speech synthesis as it is available today does not sound natural, however, and is sometimes hard to understand. Hence it is generally not recommended for use in an automatic inquiry system. Instead, we recorded all necessary phrases and words, e.g. the station names, numbers, and names of months and weekdays, and stored them in digital form on hard disk. Whenever output is to be created, the appropriate segments are concatenated and replayed.

While this approach is fairly simple and somewhat limited in flexibility, the generated speech sounds rather natural and seems to be widely accepted.

## 7. FIELD TEST AND EVALUATION

For the acquisition of training material for the recognition and understanding parts, as well as for evaluation of the dialogue and the system's overall performance, we had to make people call our system. People, however, tend to behave completely differently when simply trying out a system because they are asked to do so instead of really wanting to get an information. Therefore, it was vital for us to conduct a field test in order to obtain realistic data.

This field test was organized as a bootstrapping process. Initially, the system was trained with just the developers' voices, then the telephone number was circulated within the department, the company, and finally, the outside world. After each step, the newly collected material was harnessed for retraining and general improvements.

Beginning in February 1994, the system has also been promoted through press releases and radio interviews. The number of incoming calls showed considerable peaks after each publication and currently averages about 1000 per month.

These calls are recorded and transcribed. Approximately 30% of them cannot be used for evaluation purposes since they were made by people who only played with the system, used it for party entertainment, or hung up after the initial announcement.

The others apparently consist of real requests for connections. Of these calls, about 75% were successfully completed. One quarter of the remaining calls failed due to poor recognition performance, which we hope to improve further as we collect more training data. The rest was asking for stations that are not in the vocabulary, or had other problems with the dialogue. We are confident that we can achieve a success rate of 90% within a year.

## REFERENCES

- [1] H.Aust, M.Oerder: *Generierung einer Datenbankanfrage aus einem gesprochenen Satz mit einer stochastischen attribuierten Grammatik*. To appear in *Proc. MUSTERERKENNUNG 94*, Springer-Verlag, 1994.
- [2] H.Aust, M.Oerder, F.Seide: *Experience with an Automatic Train Timetable Information System*. To appear in *Proc. IVTTA 94*, Kyoto, 1994.
- [3] M.Bates et al.: *The BBN/HARC Spoken Language Understanding System*. In *Proc. ICASSP 93*, pp. II-111-II-114, Minneapolis, MN, USA, 1993.
- [4] H.Bergmann et al.: *Dual Use of Syntactic Information for Acoustic Recognition and Semantic Processing in a Spoken-Language Database Query System*. In *Proc. KONVENS 92*, pp. 39-48, Springer-Verlag, Berlin Heidelberg, 1992.
- [5] W.Eckert et al.: *EVAR: Ein sprachverstehendes Dialogsystem*. In *Proc. KONVENS 92*, pp. 49-58, Springer-Verlag, Berlin Heidelberg, 1992.
- [6] K.S.Fu: *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [7] F.Jelinek, J.D.Lafferty, R.L.Mercer: *Basic Methods of Probabilistic Context Free Grammars*. In *Speech Recognition and Understanding*, pp. 345-360, NATO ASI Series F, Springer-Verlag, Berlin Heidelberg, 1992.
- [8] E.P.Giachin: *Automatic Training of Stochastic Finite-State Language Models for Speech Understanding*. In *Proc. ICASSP 92*, pp. I-173-I-176, San Francisco, CA, 1992.
- [9] H.Ney, R.Haeb-Umbach, B.-H.Tran, M.Oerder: *Improvements in Beam Search for 10000-Word Continuous Speech Recognition*. In *Proc. ICASSP 92*, pp. I-9-I-12, San Francisco, CA, 1992.
- [10] M.Oerder, H.Ney: *Word Graphs: An Efficient Interface between Continuous-Speech Recognition and Language Understanding*. In *Proc. ICASSP 93*, pp. II-119-II-122, Minneapolis, MN, 1993.
- [11] J.Peckham: *A New Generation of Spoken Dialogue Systems: Results and Lessons from the SUNDIAL Project*. In *Proc. EUROSPEECH 93*, pp. 1407-1412, Berlin, Deutschland, 1993.
- [12] R.Pieraccini et al.: *A Speech Understanding System Based on Statistical Representation of Semantics*. In *Proc. ICASSP 92*, pp. I-193-I-196, San Francisco, CA, USA, 1992.
- [13] V.Steinbiss et al.: *The Philips Research System for Large-Vocabulary Continuous-Speech Recognition*. In *Proc. EUROSPEECH 93*, pp. 2125-2128, Berlin, 1993.
- [14] W.Ward: *Understanding Spontaneous Speech: The Phoenix System*. In *Proc. ICASSP 91*, pp. 365-367, Toronto, 1991.
- [15] S.J.Young, C.E.Proctor: *The Design and Implementation of Dialogue Control in Voice Operated Database Inquiry Systems*. In *Computer Speech and Language 3*: pp. 329-353, 1989.