A VOICE-CONTROLLED AUTOMATIC TELEPHONE SWITCHBOARD AND DIRECTORY INFORMATION SYSTEM

A. Kellner, B. Rueber, and F. Seide
{*kellner, rueber, seide*}*@pfa.research.philips.com*
Philips GmbH Forschungslaboratorien
P.O. Box 1980, D-52021 Aachen, Germany

ABSTRACT

In this paper, we present the Philips automatic telephone switchboard and directory information system PADIS[1]. PADIS understands natural-language requests in fluently spoken German. The system offers telephone, fax, and room numbers, email addresses, private phone numbers, and direct call completion. A setup with a 500-entry database is currently in a field test in our research laboratory and has shown a success rate of 90%. This paper describes the system architecture and its components, and presents experiences as well as results from the field test.

1. INTRODUCTION

The use of automatic speech recognition for telephone switchboard systems offers a large potential for automation and increased functionality. For such an automatic system to be accepted by the user, it is crucial that everybody can use it without any prior instructions. Thus, the user must be able to talk to the system in a natural way, very much like he or she would speak to a human operator.

We have developed a speaker-independent continuous-speech prototype of an automatic telephone switchboard system to be used over the telephone. For about 500 employees and structural units of our lab, it offers telephone numbers, fax and room numbers, email addresses, and – if explicitly agreed by the subscriber – private phone numbers. In addition, it provides direct call completion to the desired party. The prototype has been used in our research lab since March 1996. Figure 1 shows an example of a PADIS dialogue.[2]

The system is based on the technology of the Philips automatic train timetable information system [1]. However, the directory-information application has some additional requirements. First of all, the system must be able to handle large databases in a flexible manner. For that, the database is now kept as a separate entity to obtain a scalable architecture in which the database can be maintained and extended without affecting the rest of the system. In addition, the constraints contained inherently in the database are exploited to improve understanding accuracy. To optimize the flow of the dialogue, the rigid form-filling scheme used so far was replaced by a more flexible dialogue control that incorporates the database.

System:	*Hello, this is PADIS. How can I help you?*
Caller:	*I'd like to talk to Mike.*
System:	*So, you want to talk to Mike Anderson?*
Caller:	*Yes, please.*
System:	*The extension 506 is busy right now. Can I do anything else for you?*
Caller:	*Oh yes. Can you give me the email address of Mary, please?*
System:	*Which Mary do you want the email address of?*
Caller:	*Uh, she is in the speech recognition group.*
System:	*So you want the email address of Mary White from the speech recognition group?*
Caller:	*Exactly.*
System:	*The email address is ...*

Figure 1: Example dialogue.

In this paper, we describe a system architecture that meets these requirements and explain its components. We also report on experiences made in the field test and show results on field-test data.

2. SYSTEM ARCHITECTURE

An automatic inquiry system requires speech recognition, language understanding, dialogue control, and speech output capabilities. In our system, these components are organized in a modular pipeline architecture (Fig. 2). In this way, the modules can be maintained and exchanged independently of each other. Unlike in the train schedule system, the database is kept separate from the other domain-specific knowledge sources such as grammar, language model, and dialogue description, permitting database update without retraining.

[1] Philips Automatic Directory Information System

[2] All examples have been translated from German. Personal data (names etc.) have been changed for obvious reasons.

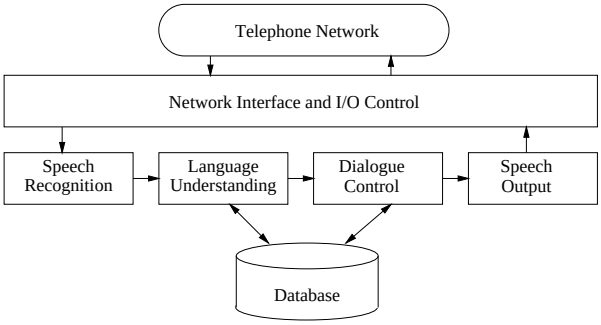

Figure 2: System architecture of PADIS.

2.1. Speech Recognition

The first module in the signal-processing pipeline is a speaker-independent continuous telephone-speech recognizer that processes spontaneous natural utterances in realtime.

The state-of-the-art continuous-density HMM recognizer uses 3502 strongly tied context-dependent phonemes that share 703 tied states. They have been trained on a large German spontaneous-speech database that we have recorded over the telephone network during a field test of our train schedule information system, comprising 33081 utterances and 12.1 hours of actual speech.

Rather than outputting the single best sentence, the recognizer creates a set of plausible sentence alternatives. The final decision is deferred to the subsequent language understanding module, permitting incorporation of additional knowledge such as understanding grammar and database.

The sentence hypotheses are represented as a compact *word graph* [2]. A word graph is a directed acyclic graph whose nodes correspond to points in time, while its arcs represent plausible word hypotheses. Each arc is assigned the word hypothesis' acoustic score. Every path through the graph is a sentence hypothesis.

Word graph generation is implemented in a two-stage process as described in [2]. The first stage, the *word-hypotheses generator*, performs a time-synchronous beam search using time-conditioned tree copies to identify and score plausible word hypotheses. The second stage, the *word-graph optimizer* combines them into a pruned word graph.

2.2. Language Understanding

The task of the language understanding is to find the most probable path through the word graph and to compute its meaning. Since we have to deal with spontaneous, natural language, the input may not always be grammatically correct. Recognition errors may also account for an input that is not completely parseable. This calls for a very robust parser that can handle erroneous or grammatically incorrect input.

For our purpose, we do not have to understand every input word, but only those phrases which contain information relevant for the database query. These meaningful phrases (so-called *concepts*) are usually well structured and are modelled by a stochastic context-free grammar [1, 3, 4]. Concepts may occur in arbitrary order in the input and may be interspersed with *filler* words that do not contain any relevant information. It is therefore sufficient to locate and further process only the concepts rather than the complete input.

Concept graph

The individual concepts are modelled by an attributed stochastic context-free grammar with a distinct start symbol for each concept. Every rule of this grammar is assigned a probability which indicates how likely it is to be applied given the left-hand side non-terminal. The grammar can thus be used as a stochastic language model.

When parsing the input, we compute its meaning at the same time. Every non-terminal can be assigned a set of attributes. For each syntactic rule for a non-terminal, there may be semantic rules to compute its attributes from the values of the attributes on the right-hand side.

Figure 3 shows some example rules of our grammar. The values in parentheses are the negative log probabilities of the corresponding rules.

```
<affiliation> ::= (1.62) in the <db_group> <dept>
        group := <db_group>.group // attribute assignment

<affiliation> ::= (3.32) of the <db_group> <dept>
        group := <db_group>.group

<dept>        ::= (1.51) group
<dept>        ::= (1.89) department

<db_group>    ::= DATABASE group  // import terminal list
        group := DATABASE.group
```

Figure 3: Example grammar rules.

The word lists and probabilities of database-specific non-terminals like `<db_group>` are not contained in the grammar but imported from the database, permitting to exchange the database without modifying or retraining the grammar.

Using this grammar, a *concept graph* is constructed from the word graph. It has the same nodes as the underlying word graph, and its edges are concept instances found by the parser. Each edge is assigned a score which consists of the acoustic score of the words contained in the concept phrase plus a language model score assigned by the grammar.

Since in general not the whole input is covered by the grammar, we add *filler arcs* to model uncovered word sequences. The language-model scores for filler arcs are computed using a word bigram model. Since the parser may also have found concept phrases that were not actually spoken by the user, a competing filler arc is also added for each concept arc.

The language model score for a whole sentence is computed by combining the language model scores for all concepts and

fillers on the path with a concept-level bigram model that models the overall sentence structure.

Integrating the database

To improve understanding accuracy, the decision on the best path through the concept graph is not only based on acoustic and language-model scores, but also incorporates database constraints and dialogue history. Actually, we explicitly model the a-priori distribution of a given information item set. In particular, a set of information items which is not consistent with the database (e.g. an invalid combination of first and last name) or with the information given in previous turns is assigned a low probability value (actually, it is set zero in our current implementation).

We use the following algorithm: First, we compute the best path through the concept graph according to the acoustic and language model. If this path turns out to be inconsistent, the next best sentence is computed using the algorithm described in [5] until a consistent path is found. By this approach, understanding errors could be reduced by relatively 27%, see below. A detailed description of this method and the underlying stochastic model can be found in [6].

2.3. Dialogue Control

The overall task of the system is to gather all information items required to uniquely identify a database query. Usually, the user does not give all needed information items in a single utterance, and there must also be a way for correcting possible understanding errors. Thus, we need an interactive dialogue with the user.

The *dialogue control module* has to fill information *slots* in the query pattern (like first and last name) with the values given by the user and it has to come up with further questions to obtain missing slot values. To obtain a more human-like dialogue, we want to avoid a rigid question-answer scheme. Instead, our system permits a mixed-initiative dialogue, i.e. the user can give the information in arbitrary order. He can also provide more or different information than he was actually prompted for. In fact, this may be even necessary if he does not know some of the items the system asked for.

To avoid giving a user wrong information due to misrecognitions, all recognized information items must be verified. We use the scheme of *implicit verification*. In every question asked by the system, the information understood in the previous turn is mentioned in the current question to permit the user to detect misunderstandings (and to correct them). Only the last question of a dialogue is a pure confirmation (cf. example in Fig. 1).

A very effective way to meet these requirements and to keep the dialogue description flexible with respect to other applications, is to use a declarative dialogue description as described in [7].

In contrast to our train schedule information system, the set of slots to be filled to refer to a unique database entry may vary in the directory information task. E.g., some first names may be unique, whereas others require further disambiguation. Therefore, the dialogue control module consults the database with the current information to determine if further information has to be asked for (cf. the example dialogue in Fig. 1, in which the first name *Mike* is already unique, whereas *Mary* is not).

In some questions, it turned out to be necessary to include additional information from the database in the confirmation prompt even if it was not provided by the user. E.g. rather than saying *"So you want to speak to White ?"*, we use *"to Mrs. White"* instead. However, this feature must be used with care: We observed that users get confused easily if the system comes up with something the user has not said, especially if it is wrong due to a recognition error.

2.4. Speech Output

For the speech output, pre-recorded phrases are concatenated. This sounds much more natural than today's state-of-the-art speech synthesis systems. However, in general this approach is unfeasible in the case of very large and/or dynamically changing databases, because for every new entry in the database, the corresponding phrases must be recorded (by the same speaker).

3. FIELD TEST EXPERIENCES

The PADIS system has been in a field test in our lab since March 1996. Since then, about 5000 dialogues have been collected corresponding to an amount of approximately 6 hours of user speech.

3.1. System Performance

In our current setup, 90% of the dialogues have been successful, i.e. in 90% of all calls, the user finally got the desired service. The average dialogue consists of 2.7 turns (with 2.7 words per turn). Since the shortest possible dialogue already takes two turns (*"I want Mike."* – *"Was that Mike?"* – *"Yes."*), this means less than one correction per dialogue. Table 1 summarizes the word and attribute error rates, which measure insertions, deletions, and substitutions for words and attributes (information items), respectively.

Table 1: Word and attribute error rates.

Setup	WER	AER
without database	28.9%	40.5%
with database constraints	24.4%	29.5%

In interpreting the error rates, it should be noted that a dialogue can be successful even if some attribute is misrecognized, e.g. a deletion of a first name when the last name is unique. The gain obtained by employing the database constraints is much higher for attribute errors than for word errors because misrecognitions of meaningless filler words are much less affected.

3.2. User Experiences

Although PADIS can also provide room numbers and email addresses, it has mainly been used for call completion (91% of all calls). A number of users already use PADIS regularly for their everyday business as a convenient replacement of the written telephone lists and hand-dialling.

How callers speak to our system

For 3411 dialogues, we examined the first dialogue utterance, in which we assumed the caller to talk most likely in his preferred way. In 64%, the user gave the first and the last name, while in 28% and 8%, the caller specified the last or first name only, respectively.

One might expect humans talking to a machine to use the simplest possible formulation, which would be the first name (or form of address) followed by the last name for connection requests. Interestingly however, we have found this formulation in only 24% of the first dialogue utterances. Most users preferred much more talkative formulations: The average length of the first dialogue utterances is 4.2 words. In over 35%, *please* was used. 31% of the utterances contain phrases like *"I'd like to"*. Finally, in 6% of the dialogues, the user began with a greeting like *hello*.

Although these figures might be somewhat biased due to callers who tested our system, it seems that if users are not restricted by the system, they prefer to talk like they would to a human even if it is not necessary for achieving their goal.

Inconsistent information

Before we integrated the database constraints, recognition errors often led to invalid combinations of first and last names and form of address. For example, users experienced it as extremely annoying to be presented with *Mrs. Mike White*, because "the system could have known."

In some cases (e.g. when encountering heavy background talk) the system may not be able to identify any sensible interpretation at all in the word graph. In an early version, we used to select the highest scoring interpretation (still violating the database constraints) and presented the user with a corresponding clarification question. However, our experiences have shown that in this case it is better to entirely discard the utterance and reformulate the question.

System prompts and dialogue efficiency

The widest dissence on our system among the users has been about the duration and conciseness of the system prompts. Novice users liked the system to introduce itself and explain its features, while especially frequent users mainly wanted a short dialogue and thus demanded very short system prompts without lengthy explanations. (Some of them actually requested to replace the greeting phrase by a simple beep.) This would, however, confuse novices completely.

As a compromise, we used a short guiding question for the greeting (such as *"How can I help you ?"*) and kept the confirmation prompts as short as possible. To further optimize dialogue duration for call-completion requests, the system does not output the number of the desired party when transferring a call. Only if the line is busy the user is told the number instead.

Randomized prompts

Finally, a simple means of making the system more pleasant is to randomly select system prompts from a set of alternative formulations. Since we have applied this technique for the greeting, users have experienced the system as "more vivid" or "less boring". We intend to extend this principle to the most frequent prompts and confirmation questions.

4. CONCLUSIONS

In this paper we have presented the Philips automatic directory information and exchange board system PADIS. We have described the system architecture and reported experiences from a field test conducted in our research laboratory.

The central new idea in our prototype is the consequent integration of database knowledge into speech understanding and dialogue control. This significantly improved understanding accuracy and allowed for a more flexible dialogue, leading to a dialogue success rate of about 90%.

The field test has shown that incorporating the database in the search for the best path not only reduced the attribute error rate by relative 27% but also increased the user acceptance since the system did not come up with invalid name combinations. We believe that using a method of this type will be essential if a much larger database is used.

5. REFERENCES

1. H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 17(3–4), pp. 249–262, Nov. 1995.
2. M. Oerder and H. Ney. Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Proc. ICASSP*, volume II, pp. 119–122, Minneapolis, April 1993.
3. R. Pieraccini and E. Levin. Stochastic representation of semantic structure for speech understanding. In *Proc. EUROSPEECH*, pp. 383–386, Genua, 1991.
4. F. Jelinek, J. Lafferty, R. Mercer. Basic methods of probabilistic context free grammars. In *Proc. NATO ASI*, pp. 345–360, Cetraro, July 1990.
5. B.H. Tran, F. Seide, and V. Steinbiss. A word graph based n-best search in continuous speech recognition. In *Proc. ICSLP*, Philadelphia, Oct. 1996.
6. F. Seide, B. Rueber, and A. Kellner. Improving speech understanding by incorporating database constraints and dialogue history. In *Proc. ICSLP*, Philadelphia, Oct. 1996.
7. H. Aust and M. Oerder. Dialogue control in automatic inquiry systems. In *ESCA Workshop on Spoken Dialogue Systems*, pp. 121–124, Vigsø, June 1995.