The Thoughtful Elephant: Strategies for Spoken Dialog Systems

Bernd Souvignier, Andreas Kellner, Bernhard Rueber, Hauke Schramm, *Member, IEEE*, and Frank Seide, *Member, IEEE*

Abstract— **In this paper we present technology used in spoken dialog systems for applications of a wide range. They include tasks from the travel domain and automatic switchboards as well as large scale directory assistance. The overall goal in developing spoken dialog systems is to allow for a natural and flexible dialog flow similar to human–human interaction. This imposes the challenging task to recognize and interpret user input, where he/she is allowed to choose from an unrestricted vocabulary and an infinite set of possible formulations. We therefore put emphasis on strategies that make the system more robust while still maintaining a high level of naturalness and flexibility. In view of this paradigm, we found that two fundamental principles characterize many of the proposed methods: 1) to consider available sources of information as early as possible, and 2) to keep alternative hypotheses and delay the decision for a single option as long as possible.**

We describe how our system architecture caters to incorporating application specific knowledge, including, for example, database constraints, in the determination of the best sentence hypothesis for a user turn. On the next higher level, we use the dialog history to assess the plausibility of a sentence hypothesis by applying consistency checks with information items from previous user turns. In particular, we demonstrate how combination decisions over several turns can be exploited to boost the recognition performance of the system. The dialog manager can also use information on the dialog flow to dynamically modify and tune the system for the specific dialog situations. An important means to increase the "intelligence" of a spoken dialog system is to use confidence measures. We propose methods to obtain confidence measures for semantic items, whole sentences and even full N-best lists and give examples for the benefits obtained from their application. Experiences from field tests with our systems are summarized that have been found crucial for the system acceptance.

Index Terms—**Application specific knowledge, combined decisions, confidence measures, dialog history, natural language understanding, spoken dialog systems.**

I. Introduction

There is a growing demand for spoken dialog interfaces in human–machine interaction as they allow adopting many of the features of human–human interaction. One reason is, of course, that the exchange of spoken information is in many cases highly efficient. Another motivation is that a user does not have to learn complicated usage instructions but can interact in a natural and intuitive way with the system.

Thus, the global goal is to develop spoken dialog systems with a natural and flexible dialog flow in which the user is unrestricted in his/her choice of formulations and in which the system tries to interpret each utterance and to find the user's intention. A dialog of this type is called a *mixed initiative dialog*, since both the user and the system can influence the dialog flow: the user, by specifying information (even if not requested by the system explicitly), negating or correcting items from previous turns or interrupting the system; the system, by requesting specific information, formulating questions aimed at disambiguation, or asking for verification.

This type of dialog imposes several problems, e.g., an unrestricted (and thus in principle infinite) vocabulary, effects of spontaneous speech like hesitations or grammatically incorrect sentences, references over several dialog turns, a large number of possible user intentions and the need to flexibly verify the crucial information items.

Depending on the application, the mixed initiative dialog style may be restricted to a more rigid approach, e.g., by limiting the user's possibility to influence the dialog flow. In such a system he/she is restricted to giving the information requested by the system and can possibly perform some basic control over the dialog, e.g., starting all over again or entering a help facility. Such a dialog type is called *system driven*, since the user can only take little initiative himself. An even more restricted strategy uses a pure form filling approach in which the user is only allowed to specify the information item he/she is prompted for. The system may even ask the user to answer in a single word, which makes the recognition task easier and is still appropriate for certain dialog situations. As a further fall-back strategy the user can be prompted for spelling a specific piece of information.

The different types of dialog may be integrated into a single spoken dialog system, using the simpler and more restricted ones as fall-back solutions if problems occur.

The choice of dialog strategy depends heavily on the specific application task. Typical scenarios in which a mixed initiative dialog is most desirable are complex applications in which the user can choose from a large variety of possible dialog goals and may even change his/her goal during the dialog. Examples of this type are the travel domain, banking applications or automatic switchboards. Here, the information items required by the system to fulfill the user's dialog goal are often not known *a priori* and have to be determined in the process of the dialog.

On the other hand, applications like large-scale directory assistance, library catalogue queries, or stock exchange market information, still present a great challenge for the speech recognition technology as they often involve large word lists (i.e., >100 000 words) with equal *a priori* distributions. For example, in a directory assistance system it is easy to determine that a user said *"Give me the number of* `name`*"* but the name can

Manuscript received March 12, 1999; revised July 27, 1999. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. James R. Glass.

B. Souvignier, A. Kellner, B. Rueber, and H. Schramm are with Philips Research Laboratories, D-52066 Aachen, Germany (e-mail: souvi@pfa.research.philips.com).

F. Seide is with the Philips Innovation Center, Taipei, F24, Taiwan, R.O.C.

Publisher Item Identifier S 1063-6676(00)00287-X

be any in the full list of database entries. For applications of this type, a more restrictive dialog type helps to improve the recognition accuracy and also seems appropriate to the task, as the dialog goal is usually known in advance (e.g., getting a database entry by specifying a number of information items).

This paper summarizes the technology used in various dialog systems developed by Philips. Examples of these are the automatic train timetable information system TABA (see [1] for details), the automatic telephone switchboard and directory information system PADIS (see [19], [20], and [31]) and the large scale directory assistance prototype PADIS-XXL (see [17], [18], [21], and [30]). Of course, systems with comparable functionality and complexity have been developed by several groups. Prominent examples are the air travel information systems by MIT (Pegasus [39]), CMU [15], BBN [3] and SRI [7], the train information system *RailTel* [23], the multimedia service kiosk *Mask* [24] by LIMSI, the weather information system *Jupiter* by MIT [38], the *How May I Help You?* call routing system [14] by AT&T, and the large-scale directory assistance systems by CSELT [5], and AT&T (VPQ [8]).

Apart from a general overview over the technology used in our spoken dialog systems, the paper presents some research topics recently investigated. Section II gives a description of the system architecture, and Section III goes into more detail for the natural language understanding module. Here we put some emphasis on the fast and easy development of new applications and describe methods to bypass the collection and transcription of large training corpora. In Section IV, we describe the dialog flow management in our systems and suggest methods to exploit the dialog history in Section V. The benefits of confidence measures are discussed in Section VI, and we close with some remarks on experiences from field tests in Section VII.

In the process of developing our dialog technology two fundamental principles have proven to be powerful concepts and have been incorporated into the system architecture and the stochastic models at various places:

a) use available sources of information as early as possible;
b) keep alternative hypotheses as long as possible.

Thus, we may compare our system to a thoughtful elephant who considers the available information before doing his next step and who does not forget any of his alternative options.

Examples for principle a) are the incorporation of a database in the search for the best sentence hypothesis or using the dialog history to reject hypotheses contradicting information given in earlier turns. Principle b) has gone deeply into the architecture of our system, as the word graph passed to the language understanding module by the recognizer is a compact way of keeping many sentence hypotheses. Also, in connection with mutually dependent information items, it pays to keep N-best lists for the information given in the different turns. Constraints imposed by the application backend (e.g., that first and last name of a person have to arise from a single database entry) may require switching to an alternative hypothesis for one of the turns. We will return to these principles at various points of this paper.

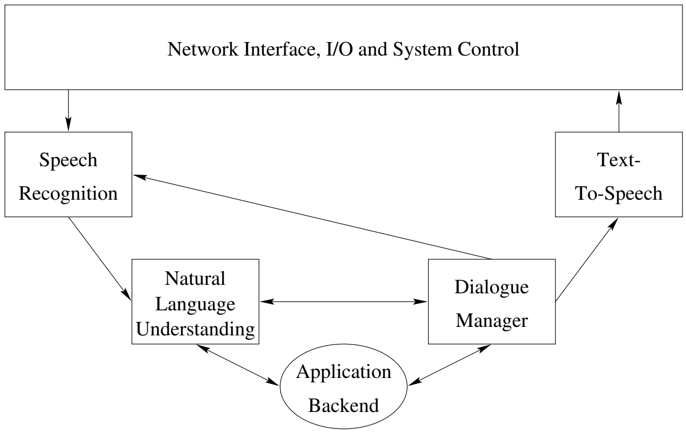

Fig. 1. Architecture for spoken dialog systems.

II. System Architecture

Fig. 1 displays the architecture of our spoken dialog systems. The main components are a speech recognizer, a natural language understanding module, a text-to-speech tool and a dialog manager.

The input data is obtained from and the output data passed to a component containing a network interface (typically telephone), I/O and top-level system control (i.e., organizing the data flow between the various components). The speech recognizer processes the acoustic data and passes a word graph to the language understanding module. For applications involving spelling, this module may contain a special spelling filter which is described below. The dialog manager integrates the understood sentence hypothesis into the system belief and decides on the necessary system actions. In particular, it passes the text string for the next system prompt to the text-to-speech module. An application backend (e.g., a database) containing task-specific information may be accessed by the language understanding module and the dialog manager.

Since we want to focus on the technology that is distinctive for spoken dialog systems, we will only briefly describe the speech recognizer, spelling filter, and text-to-speech tool in this section. The modules for natural language understanding and dialog management are discussed in more detail in Sections III and IV.

A. Speech Recognizer

The speech recognizer is a speaker-independent continuous-speech hidden Markov model (HMM) recognizer. It uses strongly tied triphones and mixtures of Gaussian densities for the state emission probabilities. For a detailed description see for example [20].

The recognizer can be modified for special tasks e.g., by using a special set of phonemes for spelling or by exploiting a single word constraint to optimize the search process (see [17] and [18]). The switching between the different recognition modes is controlled by the dialog manager.

In view of principle b) it is highly desirable to pass many alternative sentence hypotheses from the speech recognizer to the language understanding module and to delay the decision on the best hypothesis until as many sources of information as possible can be applied. Thus, instead of simply computing a single best sentence, the speech recognizer generates a word graph (cf. [27]). The nodes of this graph correspond to points in time and the arcs represent word hypotheses and are labeled with acoustic scores. The word graph is generated performing a Viterbi beam search (focused by a bigram language model) and combining for every time frame the ending words into the graph. One thus obtains a very compact representation of a large number of sentence hypotheses, since phrases occurring in several hypotheses are only stored once.

B. Spelling Filter

Since the recognition accuracy for spelled words is much higher than that for spoken words (cf. [25]), a spelling module is an important feature for spoken dialog systems. Spelling can be used as a preprocessing step to restrict the search space, as an additional source of information to identify a continuously spoken item, and of course as a problem solving strategy. For a natural language understanding system we need to cover common ways of spelling, using expressions like *"double l"* or *"f as in foxtrot"*. We have therefore included a spelling filter into our language understanding module which translates colloquial spelling phrases into sequences of letters and adds arcs with the respective letters to the spelling word graph.

Depending on the system context one can add a second step which matches the spelling word graph with a lexicon and outputs a list of lexicon entries that can be found as paths through the spelling word graph, together with their acoustic scores.

A more detailed discussion of the spelling filter can be found in [21] and [30]. In this paper we will focus on the combination of a spelling turn with spoken turns which will be discussed in Section V.

C. Speech Output

The language generation for the speech output of our system is integrated into the dialog manager, which provides templates for phrases from which the system prompts are created. These templates are filled with values from the current system belief and the resulting text is passed to the text-to-speech module.

At present, the output of the text-to-speech module is generated by concatenating prerecorded phrases. This way of speech generation sounds much more natural than today's state-of-the-art speech synthesis systems and is well-suited for applications with a moderately sized vocabulary. Its limitations are obvious, since for obtaining a naturally sounding result all phrases have to be recorded by the same speaker and, ideally, under the same conditions. This is, of course, not feasible for applications using words from a large or dynamically changing database. In these cases an automatic text-to-speech tool has to be incorporated.

III. NATURAL LANGUAGE UNDERSTANDING

As described in Section II-A, the speech recognizer passes a word graph to the natural language understanding module. In case of a spelling turn, the word graph may be processed in an intermediate step by a spelling filter as described in Section II-B before being passed to the language understanding module.

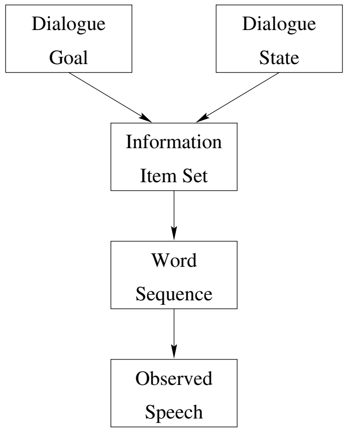

Fig. 2. Production model of a user utterance.

The language understanding module now has two tasks:

- to compute the meaning of various sentence hypotheses;
- to find and score the most probable paths through the word graph, considering all available sources of information.

Our approach to natural language understanding is based on a speech production model as shown in Fig. 2: We assume a cooperative user having a certain dialog goal (e.g., to get a specific information or to get connected to a certain person) that lies in the domain of the system and which is regarded as constant throughout the dialog. In every utterance, the user states a set of information items that is determined by the dialog goal and the current dialog state. The dialog state includes the system's current question and its current belief on what has already been stated by the user. The user casts the set of information items into a sequence of words that are finally observed by our system as a sequence of acoustic feature vectors.

In [20], a *maximum a posteriori* criterion for speech understanding is explicitly derived from this production model. The following section describes how this probabilistic framework is implemented in our language understanding architecture.

A. Language Model Set-up

The core of our language understanding module consists of three parts: a stochastic context free grammar for parsing the input and extracting the meaningful phrases (called *concepts*), a filler language model covering the parts that were not parsed, and a concept language model providing probabilities for the concept sequences. With these components path hypotheses through the word graph are rescored and at the same time endowed with their semantic interpretation. Let $W = (w_1, \cdots, w_n)$ be a sentence hypothesis. Then parsing with the grammar gives a partition $(c_1, \cdots, c_N)$ of W into a sequence of concept- and filler-phrases where each c_i is a sequence $(w_j, \cdots, w_k)$ of words from W. The parsing identifies each of these phrases as a concept C_i, where fillers can be regarded as the special concept of meaningless phrases. On the topmost layer, the concept language model assigns a probability reflecting the sequence of concepts. Since the number of concepts in one sentence is usually very small, this is realized by a bigram model (see also [12], [13], and [28]). We assume that the probability for a partial word sequence c_i only depends on the concept C_i and thus obtain

$$P(W) = P(C_{N+1}|C_N) \cdot \prod_{i=1}^{N} P(c_i|C_i)P(C_i|C_{i-1}). \quad (1)$$

Here, C_0 and C_{N+1}, refer to the sentence start and end, respectively.

For regular concepts, the conditional probability $P(c_i|C_i)$ is provided by the rule probabilities of the stochastic grammar: every rule is assigned a probability indicating how likely it is to be applied given its left-hand side nonterminal. Thus $P(c_i|C_i)$ is equal to the product of the probabilities of those rules used to generate c_i from C_i.

For a filler phrase c_i, the probability $P(c_i|C_i)$ is modeled by the filler language model which typically is a word-level bigram model.

This probabilistic framework is realized by transforming the word graph into a *concept graph*. The concept graph has the same nodes as the word graph but has two different kinds of arcs: The concept arcs cover the concept phrases found by the top-down chart parser in the word graph. The scores for these concept arcs consist of the acoustic scores for the word sequence covered by an arc and the rule probabilities for the grammar rules used to derive the respective concept. In addition to the concept arcs the concept graph contains one filler arc for each pair of nodes. This filler arc consists of the best word sequence between the start and end nodes, including rescoring with the filler language model.

The inclusion of the filler arcs serves a threefold purpose: First, the concept graph without the filler arcs may have gaps resulting from unparsable parts of an utterance, which would make it impossible to extract a best path. Second, we do not have to cover complicated phrases in the grammar that do not contribute to the information contents of an utterance and can thus keep the grammar simple. Third, we obtain some out-of-domain and out-of-vocabulary handling in phrases like *"Connect me to the Starship Enterprise"* where *"Connect me to"* is interpreted as the concept `connection` but the rest of the sentence is recognized as some meaningless garbage words indicating that no addressee was understood.

Note that by this implementation the lower level of the probabilistic framework described above (i.e., the conditional probabilities $P(c_i|C_i)$) is already included in the concept graph. This is an instance where we apply principle a), since we add information like the grammar rule probabilities to the word graph.

Starting from the concept graph, we use rescoring with the concept language model to find the best path [according to (1)]. A variation of the standard Viterbi search algorithm described in [35] allows to efficiently extract an N-best list of sentence hypotheses.

At the same time that the word graph is parsed top-down to find the concept phrases the meaning is computed bottom-up: Every nonterminal in the grammar can be assigned a set of *attributes*, the values of which are computed when the nonterminal is expanded using a syntactic grammar rule. For this, each syntactic rule may be accompanied by semantic rules which determine how the values of the attributes for the left-hand side nonterminal are computed from those of the right-hand side items.

The values of the attributes are integrated into the concept graph and are thus available for all paths through the graph. Again, we make information (the semantic contents of phrases) available at an early stage, according to principle a).

Summarizing our natural language understanding architecture, we want to stress that the concept of partial parsing that segments a sentence into meaningful and filler phrases provides our system with a high level of robustness. Since spoken dialog systems deal with spontaneous speech, this approach appears to be superior to linguistic methods which have to assume grammatically correct sentences.

A second important point is that the concept language model and especially the stochastic grammar contain application-specific knowledge that can be exploited to increase the performance of the language understanding module.

A third issue is that our language understanding architecture has much less critical stochastic parameters than a usual N-gram language model and can therefore be trained on fairly little material. (We regard the parameters of the filler language model as not critical, because its contribution to the overall speech understanding score is quite limited.) The following figures may serve as an illustration for the efficiency of a stochastic grammar: In the grammar of the train timetable information system TABA we have 176 rules describing the `time_of_day` concept. These rules contain only 82 different words as terminal symbols, but cover 232 166 218 different formulations. Of course, many of these formulations are very similar, but we still get 850 different bigrams out of them. By training rule probabilities instead of bigram probabilities we have thus reduced the number of stochastic parameters by a factor of five.

Still, the lack of available training material is one of the crucial problems in the development of new spoken dialog applications. In the following two sections we therefore propose methods to create and improve the language understanding module using no or only very little training material.

B. Initial Language Models

As mentioned above the stochastic grammar reflects much of the structure of an application. This knowledge can be used in various ways as shown in [16] or [34].

Typically, there is only little training material available when a new application of a spoken dialog system is developed. It is well known that in this situation class models perform better than usual word N-gram models (see, e.g., [10]). For example, in a class bigram model the probability $P(w_2|w_1)$ is computed as

$$P(w_2|w_1) = P(w_2|c(w_2)) \cdot P(c(w_2)|c(w_1)) \quad (2)$$

where $w \rightarrow c(w)$ is the mapping assigning a class to each word. The problem to find such a mapping is usually solved by clustering techniques (cf. [22], [26]) which in turn require

TABLE I
COMPARISON OF INITIAL LANGUAGE MODELS FOR TABA

Training Sentences	Word Error Rate			
	Word-LM	Class-LM	Grammar-LM	Fill-up-LM
0	21.51%	21.27%	16.95%	16.49%
100	20.37%	17.43%	15.95%	15.76%
1000	17.90%	15.38%	15.13%	14.53%

a training corpus. However, the information encoded in the grammar provides an alternative method: Some nonterminal symbols of the grammar can be expanded to many different terminal symbols (i.e., words), for example city names or information types (phone, fax, e-mail, address, etc.). All these words are collected into one class. Furthermore, some classes based on the semantics of a word (e.g., numbers or names) can easily be derived manually. Finally, each remaining word is declared to form a class on its own.

The class-based language models reduce the number of parameters that have to be trained, since words which are expected to occur in similar contexts are treated simultaneously. It is hoped that enough training material is available to robustly estimate this reduced number of parameters.

A different approach is to artificially increase the amount of training material. The grammar covers typical formulations for the application and thus implicitly contains information about expected user utterances. By inverting the parsing process one can create sentences from the grammar and obtains an artificial corpus which can be used as training material for N-gram language models. We use Monte Carlo methods to randomly choose rules at the branching points of the grammar and thus create random sentences which are covered by the grammar. If a small amount of training material is available this can be used to obtain weights for the different rules which leads to a more realistic corpus.

Still, the grammar only covers the meaningful parts of the user input and does not model the filler phrases. We therefore propose to combine the language model trained on the artificial corpus with the class-based language model into a single *fill-up* model (cf. [4]). In such a language model two (or more) models are combined in a hierarchical way: If an N-gram was not seen in the top-level model, its likelihood is derived from the model on the next level which in turn may fall back to lower levels.

Our experiments showed that a fill-up model using the language model estimated on the artificial corpus as top-level model and the class-based model as fall-back model performed best. Table I shows some rescoring results for experiments with different language models obtained from little training material. The application was the train timetable information system TABA. As expected, the class-based language model performs better than the usual word bigram model, but the model derived from the grammar gives a much better initial performance. Integrating the two approaches into a fill-up model combines the strengths of the two models and leads to the best results.

C. Online Adaptation of Language Models

In the previous section we described how initial language models can be obtained for a new application of a spoken dialog system. Once such initial models are available, one can run the system and collect real data. In many cases, however, this material will not be accessible to the system developer (for example, if privacy regulations prohibit storing the data) which means that it can not be used for supervised adaptation of the language models. To exploit the recognized material for an improvement of the system one therefore has to update the language models immediately after the recognition, i.e., one has to perform unsupervised online adaptation. Unfortunately, the system performance can (and does!) deteriorate if simply all first-best hypotheses for user utterances are used for adaptation. In [33] we therefore investigated methods to avoid the effects of error reinforcement caused by adaptation with misrecognized sentences.

An approach that turned out to be quite efficient is to use multiple sentence hypotheses from an N-best list as adaptation material. For that, each hypothesis gets a weight derived from its *a posteriori* likelihood such that the weights add up to one. The definition of the weights was inspired by the list-based confidence measure discussed in Section VI-A and is given in (3). Every sentence now contributes to the adaptation material according to its weight. The idea behind this approach is that well-understood parts of a sentence will occur in most of the hypotheses of an N-best list, whereas for misrecognitions there will usually be several alternatives. Thus, the effect of a recognition error is distributed over several competing hypotheses and does not result in a strong error reinforcement.

A different idea to improve the quality of unsupervised adaptation is to exclude badly recognized sentences from the adaptation material. Clearly, there are many possibilities to define when a sentence should be rejected and we investigated the following two:

1) reject a sentence if the word sequence is incorrect;
2) reject a sentence if the sequence of concepts is incorrect.

To assess the potential of these approaches we first worked with an ideal confidence measure having external information about the correctness of a recognition result. It turned out that 1) is too restrictive, since it biases the adaptation material towards short sentences and that 2) gives better results.

In a second step we applied real confidence measures to tag the concept sequences as correct/incorrect. Unfortunately, the experiments showed that the tagging errors have a strong negative influence on the quality of the adaptation material. However, with better tagging techniques using combinations of different confidence measures, we expect that adaptation excluding badly recognized material results in language models that perform comparable to those obtained from adaptation using N-best lists.

Table II shows results of experiments with the different adaptation methods for the train timetable information system TABA. An initial system was trained on 100 sentences and then adapted using different amounts of adaptation material as spec-

TABLE II
COMPARISON OF DIFFERENT MODES OF ADAPTATION FOR TABA

Adaptation Sentences	Attribute Error Rate			
	Supervised	First-best	N-best	Ideal Conf.
0	17.80%	17.80%	17.80%	17.80%
1000	14.64%	17.28%	15.46%	15.97%
3000	14.35%	19.13%	15.80%	15.32%
12000	14.12%	19.53%	15.89%	15.06%

ified in the first column. Since we adapted the whole speech understanding module, the performance of the resulting systems is measured in terms of the *attribute error rate* (AER), which evaluates errors in the relevant information items. For automatic inquiry systems this error rate is more significant than the word error rate, since the attributes determine whether the correct database query is performed. The second column of the table gives results for supervised adaptation, which serves as a baseline for what can maximally be achieved. The next columns display the results for the different real adaptation methods: unsupervised using every first best sentence, with N-best lists of length 10, with an ideal confidence measure accepting sentences with correct concept sequence. One observes that naïve unsupervised adaptation leads to a relative degradation of 9.7% in the attribute error rate, whereas adaptation using N-best lists gives 52% of the improvement obtained from supervised adaptation. Using the ideal confidence measure to exclude badly recognized material even gives 74% of the achievable improvement which shows the high potential of confidence measures for adaptation methods.

D. Integrating a Database

An important method to improve the understanding accuracy is to incorporate database constraints in the determination of the best sentence interpretation. Some of these constraints may even be included in the concept graph, for example if a concept `name` consists of a first and a last name. In this case, only concept arcs with valid first/last name combinations will be accepted. In general, however, consistency with the database can only be checked on the sentence level, since the information items can be distributed over different concepts. We therefore extract an N-best list of paths from the concept graph and rescore each hypothesis with an *a priori* distribution on the database: Let I be a set of information items (e.g., key/value pairs) and denote by $n(I)$ the number of database entries matching all items in I. Then an *a priori* distribution for the information items can be defined as

$$P_{DB}(I) = \begin{cases} 1 & \text{if } n(I) > 0 \\ 0 & \text{if } n(I) = 0 \end{cases}$$

which gives pure consistency of I with the database or as

$$P_{DB}(I) = n(I)/n(\emptyset)$$

giving the relative frequency of matches in the database. If we denote the set of information items for a sentence hypothesis W by $I(W)$, we thus replace the *a priori* probability $P(W)$ by $P(W) \cdot P_{DB}(I(W))$, since according to our speech production model displayed in Fig. 2 we assume that the cooperative user requests a consistent set of information items.

TABLE III
EFFECTS OF DATABASE RESCORING ON PADIS

Search Method	Word Error Rate	Attribute Error Rate	Avg. Position Selected
First-best	28.9%	40.5%	(1.0)
+ Database Rescoring	24.6%	31.0%	3.2
Relative Improvement	14.9%	21.0%	

Table III gives a few figures for the improvements obtained from integrating the database into our automatic switchboard system PADIS. The first row gives results for the system extracting the best path from the concept graph using only the concept language model. For the results in the second row, database rescoring was performed on an N-best list to find the best sentence hypothesis, and the relative improvements obtained from this rescoring are displayed in the third row. The columns display word and attribute error rates and the average position of the selected hypothesis in the N-best list.

IV. DIALOG MANAGER

The long-term goal in developing spoken dialog systems is to allow for a natural and flexible dialog flow which is adaptive to the user utterances.

The dialog manager has to monitor the dialog flow, collect the information given by the user, interact with the application backend, and to decide whether (and which) further information is required or whether an action is to be performed (e.g., resetting the system).

A. Strategy

The general strategy followed in our systems is termed as *slot-filling*. A slot is a certain information item for which a value is required. In a system driven dialog only values for specific slots are accepted, whereas in a mixed initiative dialog the user is allowed to give as much information as he/she wants in one turn, possibly values for slots not present in the system prompt. The task of the dialog manager is to fill enough slots to meet the user's dialog goal while keeping the dialog as short as possible. For example, in an automatic directory assistance system a typical dialog goal is to find out the phone number of a specific person. From an utterance like *"Give me the number of Mister Bean"* the system is able to assign the value *phone number* to the slot `request_type` and the value *Mister Bean* to the slot `name`. If there is only a single database entry with this name, no further information is required and the requested number is played to the user. Otherwise, the given information is not sufficient to meet the dialog goal, in which case the dialog manager tries to fill more slots with values in order to refine the database query. The database entries obtained from the last query may be consulted to determine the slot with the highest disambiguation potential which helps to avoid unnecessary dialog turns.

This is another instance where principle a) is applied, since we consult the database after each user turn to determine whether further information is required at all and, if so, which item should be requested.

An important issue for the robustness of the dialog is the usage of verification techniques. The safest method is to explicitly ask the user for verification of every single slot value. But this leads to long and fairly unnatural dialogs. A more refined approach is called *implicit verification*. For that the slot values obtained from the last user turn are included in the next system prompt asking for further information (e.g., *"When do you want to go from London to Dover?"*). If no correction is made by the user, the slot values are regarded as verified. Note, however, that users are not always aware of the possibility of correcting a system and accept prompts even if they do not match their goal. For critical actions (e.g., call transfer in a switchboard system) one may want to use explicit verification in any case.

It is obvious that already with a small number of slots to fill, specification of all possible combinations of empty, unverified and verified slots and the corresponding dialog action is not feasible. Therefore, a special dialog description language called HDDL (see [2]) was developed in which the task-specific aspects like slot definitions, questions, and verification strategies can be specified in a declarative way. The dialog module itself incorporates the general knowledge on how to conduct a slot-filling type dialog. This division of the dialog manager into a general part and a programmable task specific part allows for easy development of new applications.

B. Language Resource Management

Grammar and database provide a strong backbone for a spoken dialog system. They incorporate application specific information and make it available for the various modules in the system. They can even be exploited for the general system set-up. For example, one may use the grammar and the database to automatically generate a vocabulary tailored for an application. For that, one extracts the terminal symbols from the grammar and the words occurring in the database entries and adds them to a small background list of frequent words covering typical phrases (filler words). The following example illustrates this strategy: in our TABA train timetable system we extract 1167 station names from the database, and 443 more words occurring as terminal symbols in the grammar. This gives a coverage of 94.1% of a 23 h training text. If we add another 100 common words not found in the grammar (e.g., *also, but, or* etc.) we even reach a coverage of 97.1%. The resulting lexicon size of 1710 words is still very reasonable, which helps to get a high performance of the recognizer.

In the process of a dialog, situations can occur in which the general set-up of the system should be modified in order to optimize the system performance. The dialog description language therefore provides commands to switch the language resources (lexica, language models) used in the speech recognizer and in the natural language understanding module.

An obvious application are turns where only a small vocabulary is required, for example explicit verification or spelling turns. Moreover, we already mentioned in Section II-B that after a spelling turn only those entries of the lexicon may be activated which match the spelled information.

TABLE IV
CONTEXT DEPENDENT CONCEPT LANGUAGE MODELS FOR SWISS RAILWAY CORPUS

Language Model Type	Attribute Error Rate	Concept Error Rate
Context Independent	34.83%	17.28%
Context Dependent	31.46%	15.56%
Relative Improvement	9.7%	10.0%

A further situation in which the lexicon size can be decreased drastically is the combination of several turns in connection with a database. As will be described in Section V below, the set of active database entries is restricted from turn to turn and the vocabulary for the next turn only consists of the words occurring in the set of active database entries as values for the respective information item.

Apart from the lexicon the language models may also be switched depending on the dialog state. This allows to handle the different style of speaking observed in different dialog situations. In particular, if a very flexible dialog style falls back to a more restricted style, the language models should be modified accordingly.

In a more refined approach one may even choose context dependent language models depending on the system states, since the prompt resulting from the system state has a strongly predictive effect on the user utterance. Of course there is usually not enough training material available to estimate language models for each possible system state, but it was shown in [11] that automatic clustering techniques can be effectively used to group different system states together. Results obtained on a "real life" corpus of the Swiss Railway (SBB) are summarized in Table IV. In these experiments six automatically created clusters of system states were used to train context dependent concept language models. In the evaluation, for each dialog turn the system state was determined and the concept language model trained on the corresponding cluster was applied to determine the best path through the concept graph. Even though the rest of the system (recognizer, grammar, filler language model) remained unchanged, the application of the context dependent models resulted in a relative improvement of 10% in both the attribute error rate and the concept error rate.

V. USING THE DIALOG HISTORY

It is one of the characteristics of a dialog that information is built up over several turns and that a later utterance may contain references to an earlier one. It is thus highly beneficial to exploit the previous user turns and system prompts for the interpretation of a user turn.

In our dialog systems, we use the dialog history in various ways, from consistency checks within one turn to dynamically switching lexica and language models.

The current system status consists of one or more hypotheses for the slot values together with information about which values have already been verified. This status is called the *system belief*.

In many dialog situations, references to the system belief are implicitly made by the user. For example, after requesting a train connection a user might ask about the return trip. In this case the system simply has to swap the values for the `origin` and `destination` slots and does not have to prompt the user for this information. An example from the switchboard domain is that after requesting a person's phone number a user asks *"Give me his e-mail"*. In that case the system should treat the word *his* as if its present values for the slots specifying the person had been uttered. Thus, it is important to specify points in the dialog at which slots are cleared and to otherwise keep the slot values in the system belief. This is catered for by the dialog description language HDDL which provides commands for the slot handling (i.e., keeping and resetting) in the application specific part of the dialog manager.

A way to implement the influence of the system belief (and thereby the dialog history) on the interpretation of a user utterance, is to apply some consistency checks to reject implausible hypotheses. Depending on the application, these consistency checks can be any of the following.

- *Consistency within the turn*: The utterance may not contain two different values for the same slot, unless one of them is negated. Also, utterances with values for different slots that have no match in a consulted database are rejected.
- *Consistency with system prompt*: Corrections to items not occurring in the system prompt are not accepted.
- *Consistency with system belief*: Interpretations that do not refer to a valid database entry after being combined with the current system belief are rejected.

In our probabilistic framework as described in Sections III-A and III-D these consistency checks are integrated as a further rescoring step similar to the rescoring with the database distribution. In practice, these rescoring steps are of course performed simultaneously on the N-best list extracted from the concept graph, which gives a very efficient computational behavior of the algorithm determining the best hypothesis. A detailed description of the decision rule for the best sentence can be found in [20] and [31].

Ideally [and following our principle b)] one would keep multiple system beliefs and would combine hypotheses for new information with the different system beliefs, keeping only the consistent combinations. This has been theoretically investigated in [36] and as an application the handling of negative knowledge was introduced in the automatic switchboard system PADIS. Apart from keeping (positive) values for each slot, also values which are negated for this slot are stored. These negative slot values require a special treatment, as they are not made transparent to the user. In particular, they are only accepted if the recognition is very reliable, since they are not verified (questions like *"So, you really don't want to talk to Mr. Bean?"* are not particularly useful for a natural dialog flow). This technique of considering negative knowledge helps to avoid annoying vicious circles in the dialog flow.

The problem of changing the system belief can be solved much more easily in system driven dialog applications where the set of slots to be filled is fixed. In this situation, multiple system beliefs can be realized by keeping N-best lists holding values from different dialog turns and by combining these lists as described in [17], [18], and [30]. This technique is used in our large-scale directory assistance prototype PADIS-XXL. Here, N-best lists are kept for the values extracted from the turns in which the user speaks the last name, first name, and street name and, optionally, where he/she spells (part of) the last name. After each turn, the obtained values are used to restrict the active database to those entries for which all values are contained in the N-best lists. As already mentioned in Section IV-B this gives at the same time the possibility to restrict the recognizer's vocabulary, as only values from the active subset of the database have to be activated for the next dialog turn. The database entries get scores that are (weighted) linear combinations of the scores for the values obtained in the different turns, and the best database entry with respect to this score is returned.

Our PADIS-XXL prototype deals with the city of Aachen, Germany, which has 131 000 database entries. The lexicon consists of 38 606 last names, 9950 first names, and 2294 street names (cf. [21], [30]).

We start with recognizing the spoken first name and the pruning restricts the lexicon to an N-best list containing the best hypotheses. We then consult the database to obtain the last names matching one of the first names in the N-best list. The next recognition turn now restricts the list of last names to an N-best list of hypotheses and we consult the database again to obtain the street names matching first/last name combinations from the two N-best lists. The final recognition turn then returns an N-best list for the street names and we determine the database entry with the best combined score. The combined scores for the database entries are usually very distinctive, especially the score distance between the first and second best is quite significant. Almost all cases where the first best database entry is not correct, are due to pruning errors (graph errors) where in one turn the correct word is not contained in the entire N-best list.

In a slightly different scenario, we performed offline experiments on the telephone directory of Berlin, Germany, which contains 1.3 million entries. We optionally start with spelling the last name and then have turns for speaking the last, first, and the street names. As described in Section II-B the spelling turn is used to build a letter graph which is afterwards matched with the word list of last names to restrict the lexicon for the spoken last names.

The hierarchical combination scenario corresponding to our PADIS-XXL prototype is summarized in Tables V and VI. The lexicon sizes given are average values over 676 dialogs. The relatively high error rate of 25.00% at an average lexicon size of 10.3 words for the spoken last names after the spelling turn is quite irrelevant, since in effect the recognizer performs a rescoring of the word hypotheses obtained from the spelling filter without discarding any hypothesis. This can be seen from

TABLE V
HIERARCHICAL RECOGNITION SCENARIO WITH SPELLING (BERLIN)

Turn	Average Lexicon Size	Turn Error Rate	Combination Error Rate
Spelling		21.15%	21.15%
+ Last Name	10.3	25.00%	16.42%
+ First Name	183.2	13.46%	12.57%
+ Street Name	202.6	11.69%	8.88%

TABLE VI
HIERARCHICAL RECOGNITION SCENARIO WITHOUT SPELLING (BERLIN)

Turn	Average Lexicon Size	Turn Error Rate	Combination Error Rate
Last Name	189,352	65.68%	65.68%
+ First Name	5,443	31.66%	30.18%
+ Street Name	3,609	25.00%	9.17%

the fact that the graph error rates of 6.5% are identical for these two turns. This relatively high graph error rate for the spelling turn also explains why the combination result with spelling is only slightly better than that without spelling.

It is interesting to note that in the combinations using the spelling turn 2.66% of the resulting combination lists are actually empty, which is of course a safe criterion to fall back to a human operator. One thus has only 6.22% undetected misrecognitions.

In addition to the hierarchical strategy followed in our PADIS-XXL prototype, we also combined the N-best lists of separate recognition turns using the full lexicon for the respective turns. Again, also the effect of omitting the spelling turn was investigated.

Table VII shows results for the separate recognitions and their combinations. One observes that the spelling turn does in fact increase the error rate of the full combination. However, the benefit of the additional turn is that due to the redundant information 12.28% of the combination lists are empty and one therefore has a remarkably low 2.22% of undetected misrecognitions while in the set-up without spelling there are no empty lists at all.

VI. CONFIDENCE MEASURES

As automatic speech recognition can not deliver error-free results, it is important to judge the quality of a recognition result in order to detect possible misrecognitions. This is in particular true for spoken dialog systems, since incorrectly understood information items lead to an erroneous system behaviour (e.g., a wrong database query) and often results in severe user frustration, as the dialog goal can not be reached. On the other hand, the detection of recognition problems can be used by the dialog manager either by starting a clarification subdialog or by falling back to a more robust dialog style.

TABLE VII
SEPARATE RECOGNITION SCENARIO (FULL VOCABULARY IN EACH TURN) WITH SUBSEQUENT COMBINATION OF N-BEST LISTS (BERLIN)

Turn	Lexicon Size	Turn ER	Combination ER	
			Spelling	No Spelling
Spelling		21.15%	21.15%	–
+ Last Name	189,352	65.68%	18.49%	65.68%
+ First Name	52,408	61.98%	15.38%	30.47%
+ Street Name	9,130	26.78%	14.50%	10.36%

A. *Confidence Measures for Semantic Items*

The crucial items for the interpretation of a user utterance are the information bearing phrases (which are covered by the concept arcs in the concept graph). In order to assess the accuracy of such an information bearing phrase one might combine confidence measures for the words constituting the phrase. But as shown in the previous sections and according to our principle a), it pays to use all available knowledge sources to extract the best path hypothesis, and some of these sources are not available on word level (e.g., grammar rule probabilities or concept language model). Building on ideas from [9] and [37] we therefore developed a confidence measure based on sentence probabilities that allows us to directly obtain reliability information for the semantic items found in an N-best list of hypotheses for the user utterance (see [29]). For that we normalize the probability mass in the N-best list to one and define the confidence measure of a semantic item as the sum of the probabilities of the sentences in which it occurs. Denote the score of the ith hypothesis in the N-best list by sc_i and let $I_A \subset \{1, \cdots, N\}$ be the set of indices of those hypotheses in which the attribute A occurs, then we define a confidence $c(A)$ for A as

$$c(A) := \frac{\sum_{i \in I_A} \exp(-\lambda sc_i)}{\sum_{j=1}^{N} \exp(-\lambda sc_j)}. \tag{3}$$

The scaling factor λ was introduced since the scores delivered by the recognizer are *scaled* negative logarithms of probabilities. It turned out that λ is useful as a tunable parameter which controls how the probability mass is distributed over the N-best list. For $\lambda = 0$ the hypotheses in the N-best list get the same weight and choosing $\lambda \gg 1$ shifts emphasis to the first best hypothesis.

It is obvious that this method is not restricted to semantic items but can easily be applied to sequences of words or full utterances.

A common way to assess confidence measures are receiver operating characteristics (ROC's) which show recognition accuracy versus false-alarm rates. Fig. 3 shows ROC-curves for different numbers of hypotheses considered in the N-best list. It displays in particular the special case that only two hypotheses are considered, which reduces our approach to the standard "second-best" method using likelihood ratios between the first- and second-best path (cf. [32]). One observes that increasing the number of hypotheses considered results in a performance gain of the confidence measure and that, especially for obtain-

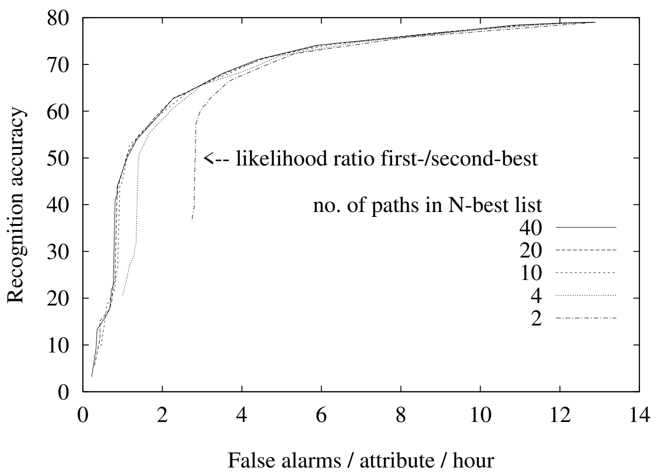

Fig. 3. Receiver operating characteristics for semantic items in PADIS

ing low false-alarm rates our method clearly outperforms the "second-best" method.

The proposed confidence measure has a variety of applications. A standard problem is to detect out-of-vocabulary (OOV) words or utterances and to start a suitable clarification subdialog.

Another example is the dialog manager's verification strategy, which may depend on the reliability of the recognized attribute. Above some high threshold, the recognition is regarded as safe and no verification is performed. Above a lower threshold, the dialog manager might use implicit verification and below that explicit verification or rejection. This leads to a more efficient dialog flow as verification turns are omitted for well-recognized items and it has successfully been incorporated into our automatic switchboard system PADIS (see [6] for a more detailed discussion). The reliability thresholds by which the verification strategy is chosen may even be made dependent on the cost of a wrong action caused by a misrecognition. For example, in a system-driven dialog, the commands by which the user can influence the dialog flow result in critical actions and should only be accepted above a fairly high threshold. This concept has been realized in our automatic directory assistance prototype PADIS-XXL, where the user may restart the dialog, enter a help facility or request being connected to a human operator. It is highly desirable not to perform any of these actions accidentally.

An issue related to the verification strategy is to decide on fall-back methods (e.g., to a simpler recognition scenario or to a human operator).

A different type of application for confidence measures was already mentioned in Section III-C. Here the confidence measure is used to decide whether the quality of a recognition result is high enough to be used for online adaptation.

B. *Confidence Measures for N-best Lists*

In view of applications where hypotheses from N-best lists are combined, one has to judge whether an N-best list is useful in a combination decision or not. Since the combination with a different turn may lift a hypothesis from some position in the N-best list to the top, usability in this sense is not determined by the likelihood that the first-best hypothesis is correct but by the likelihood with which the correct hypothesis can be lifted to the top in a combined decision. In contrast to the previous section, one therefore requires a confidence measure for a whole list rather than for a single hypothesis from the list.

The method proposed in [18] is a variation of the list-based approach described in the previous section. Instead of accumulating probabilities for hypotheses containing a chosen attribute we simply accumulate the *a posteriori* probabilities for the first best sentences up to a certain break criterion. This criterion can be a fixed number of hypotheses or a score difference to the first best sentence. A simpler approach than accumulating the probabilities is to just look at the size of the chosen set of first best hypotheses in relation to the size of the full N-best list. This is the special case of choosing the scaling factor λ in (3) as $\lambda = 0$. It turned out that the criterion using the score difference to the first best sentence works well and that both the accumulated probabilities and the relative frequencies give satisfactory results. At an operating point where only 4.9% of not useful lists are accepted, 61.5% of the useful lists are accepted, which is a relative gain of 57% over a random tagging.

The application of this confidence measure is to reject N-best lists below a certain reliability threshold, since they appear to be not useful for a combination decision. Depending on the application, one may prompt the user to give the same information again in order to obtain a new N-best list of (hopefully) better quality, fall back to a more robust recognition scenario (e.g., spelling), or transfer the user to a human operator.

VII. Usability Studies

During the field tests with the different dialog systems, we obtained some interesting feedback which helped to improve the systems and to increase the user acceptance and satisfaction.

A striking (and from a technology-oriented point of view disappointing) experience was that people judged the system mainly by the speech output and did not regard speech recognition as a difficulty at all. Our approach to generate the speech output by concatenating prerecorded words and phrases was well accepted by the users. But as pointed out in Section II-C, this method is limited to applications with a moderately sized vocabulary and has to be replaced by a speech synthesis system in case of a large or dynamically changing vocabulary.

A simple means of making the system more pleasant is to randomly select system prompts from a list of alternatives. After introducing a randomized greeting prompt in our PADIS system the users described the system as "more vivid" and "less boring".

An important observation was the wide range of user preferences, e.g., for the duration and conciseness of the system prompts. While novice users liked explanatory system prompts, more experienced users tended towards a shorter dialog and some even requested a simple beep instead of the greeting prompt. This diversity suggests the introduction of user models influencing the dialog management.

An interesting aspect coming up in the field tests was the style of speaking chosen by the users. In general we observed a fairly "talkative" way of communication with the system and not so much a "machine-like" style. The latter usually comes up after recognition errors, since users may not be familiar with the concept of implicit verification. For example, the system prompt *"When do you want to travel to Denver"* resulting from a misunderstood travel destination is not corrected by saying *"I want to go to Dallas"* or *"I don't want to go to Denver"* but by a simple *"No"*.

Of course, the user's style of talking is heavily influenced by the system prompts. Specific prompts like *"From where to where do you want to travel"* trigger well predictable answers, whereas replies to *"What can I do for you?"* have a large diversity. We also observed that users tend to reuse phrases from the system prompt rather than reformulating them.

This leads to a more general aspect of system design. A user has what may be called a *cognitive model* (cf. [6]) of the system which reflects what he/she regards as the current system belief. Part of this cognitive model is the utterance given to the system, since the user expects it to understand its input. If the system does not react in accordance with the user's cognitive model, he/she often gets confused and problems in the dialog flow are likely to occur. The main reason for a discrepancy of this kind is of course that the system belief may contain misrecognized items which can not be part of the cognitive model as they are unknown to the user. It is therefore important that the user is aware of the possibility of recognition errors. But also the user's formulations form part of the cognitive model, hence the system should try to pick up phrases from the user utterances where possible instead of using internally fixed formulations (i.e., it should avoid prompting the user with *"4.15pm"* in reply to *"a quarter past 4 in the afternoon"*). Another reason why the system belief may differ from the cognitive model is that information can be obtained from sources other than the user utterance (e.g., a database). In this case, the user might be surprised being confronted with items he/she did not specify at all.

To avoid conflicts between the user's cognitive model of the system and the current system belief, it is important to make the latter as transparent as possible, which of course has to be balanced with the efficiency of the dialog flow. This means to find a compromise between system verbosity, reliability and dialog duration.

VIII. Conclusion

We have presented strategies for spoken dialog systems in various application domains. The overall goal is to allow for a natural and flexible dialog flow while on the other hand obtaining a high level of robustness of the system. We have demonstrated that in order to approach this goal it is very beneficial to consider available sources of information as early as possible and to keep alternative hypotheses as long as possible. The realization of these basic principles was illustrated for different aspects of spoken dialog systems, from the word graph as interface between speech recognizer and language understanding module, over plausibility checks considering background information from a database to combination decisions over several user turns.

We also showed that the architecture of our natural language understanding module allows us to develop new applications with no or very little training material and we discussed how confidence measures can be applied to improve the quality of a dialog system.

The proposed methods have successfully been incorporated into dialog systems for applications from different domains (timetable information, telephone switchboard, directory assistance) and have proven to increase the performance and usability of the systems.

References

[1] H. Aust, M. Oerder, F. Seide, and V. Steinbiss, "The Philips automatic train timetable information system," *Speech Commun.,* vol. 17, pp. 249–262, 1995.

[2] H. Aust and M. Oerder, "Dialogue control in automatic inquiry systems," in *Proc. ESCA Workshop on Spoken Dialogue Systems*, Vigsø, Denmark, 1995, pp. 121–124.

[3] M. Bates *et al.*, "The BBN/HARC spoken language understanding system," in *Proc. ICASSP'93*, vol. II, Minneapolis, MN, pp. 111-114.

[4] S. Besling, H.-G. Meier, "Language model speaker adaptation," in *Proc. EuroSpeech'95*, Madrid, Spain, pp. 1755–1758.

[5] R. Billi, F. Canavesio, C. Rullent, "Automation of Telecom Italia directory assistance service: Field trial results" in *Proc. IVTTA'98*, Torino, Italy, pp. 11–16.

[6] G. Bouwman and J. Hulstijn, "Dialogue strategy redesign with reliability measures," in *Proc. First Int. Conf. on Language Resources and Evaluation*, Granada, Spain, 1998, pp. 191–198.

[7] H. Bratt, J. Dowding, and K. Hunicke-Smith, "The SRI telephone-based ATIS system," in *Proc. ARPA Workshop on Spoken Language Technology*, Austin, TX, 1995, pp. 22–25.

[8] B. Buntschuh *et al.*, "VPQ: A spoken language interface to large scale directory information," in *Proc. ICSLP'98*, Sydney, Australia, pp. 2863–2866.

[9] L. Chase, "Word and acoustic confidence annotation for large vocabulary speech recognition," in *Proc. EuroSpeech'97*, Rhodes, Greece, pp. 815–818.

[10] A.M. Derouault and B. Merialdo, "Natural language modeling for phoneme-to-text transcription," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-8, pp. 742–749, 1986.

[11] E.W. Drenth and B. Rueber, "Context-dependent probability adaptation in speech understanding," *Comput. Speech and Lang.,* vol. 11, pp. 225–252, 1997.

[12] K.S. Fu, "Stochastic languages for syntactic pattern recognition," in *Syntactic Pattern Recognition and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1982, pp. 196–245.

[13] E.P. Giachin, "Phrase bigrams for continuous speech recognition," in *Proc. ICASSP'95*, Detroit, MI, pp. 225–228.

[14] A.L. Gorin, G. Riccardi, and J.H. Wright, "How may I help you?," *Speech Commun.,* vol. 23, pp. 113–127, 1997.

[15] S. Issar, W. Ward, "CMU's robust spoken language understanding system," in *Proc. EuroSpeech'93*, Berlin, Germany, pp. 2147–2150.

[16] A. Kellner, "Initial language models for spoken dialogue systems," in *Proc. ICASSP'98*, Seattle, WA, pp. 185–188.

[17] A. Kellner, B. Rueber, and H. Schramm, "Strategies for name recognition in automatic directory assistance systems," in *Proc. IVTTA'98*, Torino, Italy, pp. 21–26.

[18] ——, "Using combined decisions and confidence measures for name recognition in automatic directory assistance systems." in *Proc. ICSLP'98*, Sydney, Australia, pp. 2859–2862.

[19] A. Kellner, B. Rueber, and F. Seide, "A voice-controlled automatic telephone switchboard and directory information system," in *Proc. IVTTA'96*, Basking Ridge, NJ, pp. 117–120.

[20] A. Kellner, B. Rueber, F. Seide, and B.-H. Tran, "PADIS—An automatic telephone switchboard and directory information system," *Speech Commun.,* vol. 23, pp. 95–111, 1997.

[21] A. Kellner, F. Seide, and B. Rueber, "With a little help from the database," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, CA, 1997, pp. 566–574.

[22] R. Kneser and J. Peters, "Semantic clustering for adaptive language modeling," in *Proc. ICASSP'97*, Munich, Germany, pp. 779–782.

[23] L.F. Lamel *et al.*, "The LIMSI RailTel system: Field trial of a telephone service for rail travel information," *Speech Commun.,* vol. 23, pp. 67–82, 1997.

[24] L. Lamel *et al.*, "User evaluation of the Mask kiosk," in *Proc. ICSLP'98*, Sydney, Australia, pp. 2875–2878.

[25] M. Meyer and H. Hild, "Recognition of spoken and spelled proper names," in *Proc. EuroSpeech'97*, Rhodes, Greece, pp. 1579–1582.

[26] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modeling," *Comput. Speech Lang.,* vol. 8, pp. 1–38, 1994.

[27] M. Oerder and H. Ney, "Word graphs: An efficient interface between continuous-speech recognition and language understanding," in *Proc. ICASSP'93*, Minneapolis, MN, pp. 119–122.

[28] R. Pieraccini and E. Levin, "Stochastic representation of semantic structure for speech understanding," in *Proc. EuroSpeech'91*, Genoa, Italy, pp. 383–386.

[29] B. Rueber, "Obtaining confidence measures from sentence probabilities," in *Proc. EuroSpeech'97*, Rhodes, Greece, pp. 739–742.

[30] F. Seide and A. Kellner, "Towards an automated directory information system," in *Proc. EuroSpeech'97*, Rhodes, Greece, pp. 1327–1330.

[31] F. Seide, B. Rueber, and A. Kellner, "Improving speech understanding by incorporating database constraints and dialogue history," in *Proc. ICSLP'96*, Philadelphia, PA, pp. 1017–1020.

[32] A.R. Setlur, R.A. Sukkar, and J. Jacob, "Correcting recognition errors via discriminative utterance verification," in *Proc. ICSLP'96*, Philadelphia, PA, pp. 602–605.

[33] B. Souvignier and A. Kellner, "Online adaptation for language models in spoken dialogue systems," in *Proc. ICSLP'98*, Sydney, Australia, pp. 2323–2326.

[34] A. Stolcke and J. Segal, "Precise n-gram probabilities from stochastic context-free grammars," in *Proc. ACL*, 1994, pp. 74–79.

[35] B.H. Tran, F. Seide, and V. Steinbiss, "A word graph based N-best search in continuous speech recognition," in *Proc. ICSLP'96*, Philadelphia, PA, pp. 2127–2130.

[36] B. Vromans, "Using history in dialogue management of automatic inquiry system," Master's thesis, Univ. Delft, The Netherlands, 1998.

[37] M. Weintraub, "LVCSR log-likelihood ratio scoring for keyword spotting," in *Proc. ICASSP'95*, Detroit, MI, pp. 297–300.

[38] V. Zue *et al.*, "From interface to content: Translingual access and delivery of on-line information," in *Proc. EuroSpeech'97*, Rhodes, Greece, pp. 2227–2230.

[39] V. Zue *et al.*, "PEGASUS: A spoken dialogue interface for on-line air travel planning," *Speech Commun.,* vol. 15, pp. 331–340, 1994.