PADIS –
AN AUTOMATIC TELEPHONE SWITCHBOARD AND DIRECTORY INFORMATION SYSTEM

A. Kellner[1], *B. Rueber*[1], *F. Seide*[1], *and B.-H. Tran*[2]

[1]{kellner, rueber, seide}@pfa.research.philips.com
Philips GmbH Forschungslaboratorien
P.O. Box 1980, D-52021 Aachen, Germany

[2]tran@acn.be.philips.com
Philips Speech Processing – Dialogue Systems –
Kackertstr. 10, D-52072 Aachen, Germany

Abstract

The Philips automatic telephone switchboard and directory information system PADIS provides a natural-language user interface to a telephone directory database. Using speech recognition and language understanding technologies, the system offers phone numbers, fax numbers, email addresses, and room numbers as well as direct call completion to a desired party.

In this paper, we present the underlying probabilistic framework, the system architecture, and the individual modules for speech recognition, language understanding, dialogue control, and speech output. In addition, we report results on performance and user behaviour obtained from a field test in our research lab with a 600-entry database.

We derive a new maximum-*a-posteriori* decision rule which incorporates database knowledge and dialogue history as constraints in speech recognition and language understanding. It has improved speech understanding accuracy by 19% (in terms of concept error rate), and reduced attribute substitution errors (e.g. recognition of a wrong name) by 38%.

The decision rule is implemented in a multi-stage approach as a combination of state-of-the-art speech recognition, partial parsing with an attributed stochastic context-free grammar, and an N-best algorithm which is also described in this paper.

The system conducts a flexible mixed-initiative dialogue rather than using a rigid form-filling scheme, and incorporates database knowledge to optimize the dialogue flow.

Zusammenfassung

PADIS, das automatische Telefonauskunftsystem von Philips ist eine natürlichsprachliche Benutzerschnittstelle zu einer Telefondatenbank. Unter Einsatz von Spracherkennungs- und Sprachverstehenstechnologie bietet das System die Möglichkeit zur Abfrage von Telefon-, Fax-, und Zimmernummern, Email-Adressen, sowie eine direkte Rufweiterleitung zu einem gewünschten Anschluß.

In diesem Artikel präsentieren wir das zugrundeliegende statistische Framework, die Systemarchitektur, und die einzelnen Funktionsblöcke Spracherkennung, Sprachverstehen, Dialogsteuerung und Sprachausgabe. Weiterhin stellen wir Ergebnisse zu Systemperformance und Benutzerverhalten vor, die in einem Feldtest in unserem Forschungslabor mit einer 600 Einträge großen Datenbank ermittelt wurden.

Wir leiten eine neue Maximum-a-posteriori-Entscheidungsregel ab, welche sowohl das Datenbankwissen als auch den bisherigen Dialogverlauf als Randbedingungen in Spracherkennung und -verstehen einbezieht. Es ergibt sich eine Verbesserung des Sprachverstehens von 19% (gemessen als Konzeptfehlerrate), sowie eine Reduktion von Attribut-Substitutionsfehlern (etwa Fehlerkennung eines Namens) von 38%.

Die Entscheidungsregel wurde in einem mehrstufigen Ansatz implementiert, einer Kombination von Spracherkennung, partiellem Parsing mit einer attributierten, stochastischen, kontextfreien Grammatik, und einem N-best-Algorithmus, welcher ebenfalls in diesem Artikel beschrieben wird.

Das System führt einen flexiblen "Mixed-initiative"-Dialog anstatt ein starres Form-Filling-Schema zu verfolgen und verwendet Datenbankwissen zur Optimierung des Dialogflusses.

Keywords:

directory information; speech understanding; speech recognition; natural-language understanding; mixed-initiative dialogue; database constraints; dialogue history; N-best; stochastic context-free grammar; attributed grammar; user behaviour

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 prior instructions. Thus, the user should be able to talk to the system in a natural way, very much like she or he would talk to a human operator[1].

We have developed a speaker-independent continuous-speech prototype of an automatic telephone switchboard system to be used over the telephone. This automated switchboard is being field-tested in our research lab and offers telephone numbers, fax numbers, email addresses, and – if explicitly agreed by the subscriber – private phone numbers of about 600 employees and abstract units of our lab. Its main application, however, is direct call completion to the desired party. The system is designed to replace the written telephone lists that had been used before and to handle most of the internal phone calls. The prototype has been used in our lab since March 1996.
Figure 1 shows an example of a PADIS dialogue.

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.

The PADIS system is based on the Philips automatic inquiry system technology that also has been used in our train timetable information system [3]. Besides improvements in the basic technologies (speech recognition and speech understanding), new technologies were introduced to meet the additional requirements of a directory information application. The most important of these requirements is the flexible and scalable handling of large telephone databases: On the one hand, it must be possible to modify the database without affecting the rest of the system; on the other hand, the database contains a lot of additional information that can be exploited to improve speech understanding and to optimize the flow of the dialogue. To meet both these requirements, the database is kept as a separate entity which interacts with the other system components.

[1]from here on, we have decided, for simplicity's sake, the caller to be a male.

This paper is organized as follows:
Section 2 describes the overall architecture of the PADIS system. Then we introduce a probabilistic model for an interactive dialogue system which includes database knowledge and dialogue history. This model and the resulting decision rule are presented in section 3. In section 4, the speech recognizer used in the system is explained. Section 5 deals with the speech understanding component of the system and explains the implementation of database constraints. The dialogue control module is explained in section 6. After a short description of the speech output technology in section 7, section 8 reports some results and experiences from the field test.

2 System Architecture

Our automatic inquiry system consists of a speech recognizer, a natural-language understanding component, a dialogue control module, and speech output capabilities. These components are organized in a pipeline architecture with a superordinate control module that handles the telephone interface and controls the data flow between the modules (see figure 2). In this way, the individual modules can be maintained and exchanged independently of each other.

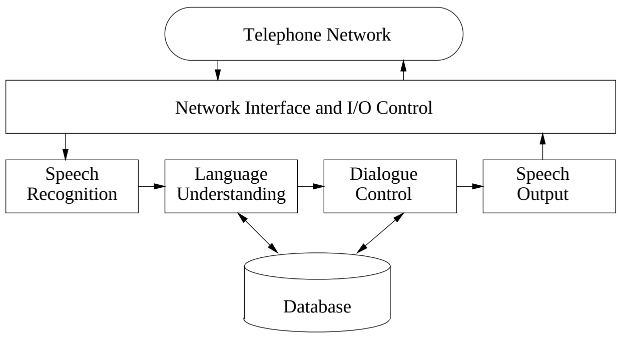

Figure 2: System architecture of PADIS.

A requirement for a practical system is that the database can be modified, updated, and maintained by a system manager who has no knowledge of a speech understanding system. Internal data files such as recognition lexica and language models should not have to be accessible to the system manager. Thus, the database is kept separate from the other domain-specific knowledge sources such as the speech-understanding grammar, language models, and dialogue description.

We use a statistical approach for the speech understanding process that integrates speech recognition and language understanding. In [18], we derived a decision rule for speech understanding which is also capable of incorporating the additional knowledge we have at hand in the form of the telephone directory database. Before we describe the four modules of our system, we will explain this probabilistic framework, which significantly influences the design of the speech recognizer and the language-understanding module.

3 Probabilistic Framework

It is well known that the error probability of a pattern recognition system can be minimized if it is based on a maximum-*a-posteriori* (MAP) probability criterion. In this section, we want to derive a MAP criterion for speech understanding that integrates all involved parts and components such as the recognizer and the telephone database.

Our system is based on a speech production model as shown in figure 3: We assume that the user has a certain dialogue goal G in talking to the system, which may be to get connected to some person or to get a telephone number, room number, or e-mail address. We further assume a cooperative user, i.e, G is not out of the domain of our system and does not change during the course of a dialogue.

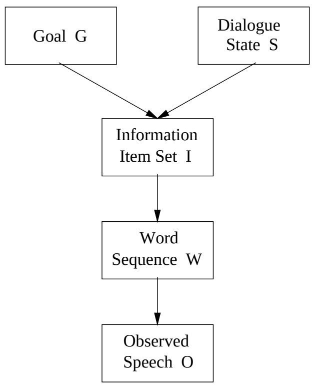

Figure 3: Production model of a user's utterance.

In every utterance, the user states a set of information items I, which is determined by his goal G and the current dialogue state S. S includes the system's current question and the system's current belief on what has already been stated by the user. From the system's point of view, the dialogue state S can be directly observed, while G is a hidden variable. I is then cast into a sequence of words W, i.e. after deciding *what* to say, the user chooses *how* to say it. The utterance is finally observed by our system as a sequence of acoustic feature vectors O. All symbols used in the following formalism are summarized in table 1.

Note that the user's goal G and the dialogue state S are modelled to be independent of each other, while indeed they are not, but rather coupled via the information collected in previous turns.

Table 1: Summary of symbols with examples.

Sym.	Explanation	Example
G	dialogue goal	{ talk to M. Jones }
G_{LM}	LM-modelled part	{ talk to }
G_{DB}	DB-modelled part	{ M. Jones }
S	system status	current belief = { Martin } question = *Which Martin ?*
W	word sequence	`give me Doctor Jones please`
W^T	word seq. templ.	`give me <title><name> please`
I	inform. item set	{ `@request=connect`, `@title=Doctor`, `@name=Jones` }
I^T	inform. template	{ `@request=connect`, `@title=<title>`, `@name=<name>` }
O	acoustic obs.	(observed feature vectors)

3.1 MAP Criterion for Speech Understanding

We define the speech understanding task as, for every utterance, finding the information item set $\hat{I}$ which most probably generated our acoustic observation O, given the system's current state S (maximum-a-posteriori criterion):

$$\begin{aligned} \hat{I} &= \arg\max_I P(I|OS) \qquad (1) \\ &= \arg\max_I p(IOS)/p(OS) \end{aligned}$$

Since $p(OS)$ is independent of I, it does not contribute to the maximization and can be omitted. Introducing the underlying word sequence W and the (unknown) dialogue goal G, we get:

$$\begin{aligned} \hat{I} &= \arg\max_I \sum_{W,G} p(OWISG) \\ &= \arg\max_I \sum_W p(O|W) \cdot \sum_G P(WISG) \end{aligned}$$

For $P(WISG)$, we assume a simple dependence between (W, I, G) and S as follows: If the hypothesized information item set I contradicts the system state S built up so far, $P(WISG)$ is simply 0, since we assume a cooperative user who does not change his goal during the course of the dialogue. (A correction of an information item is not considered a contradiction.) If on the other hand I is consistent with S, we assume no further dependence, i.e. $P(WISG) = P(WIG) \cdot P(S)$. This assumption ignores the fact that users tend to answer the question they are asked by the system (which is included in S).

With the *consistency operator* $\delta_{X,Y}$,

$$\delta_{X,Y} = \begin{cases} 1 & \text{if } X \text{ is consistent with } Y \\ 0 & \text{otherwise} \end{cases}$$

we can write $P(WISG)$ as:

$$P(WISG) \quad = \quad P(WIG) \cdot \delta_{I,S} \; \delta_{I,I} \cdot \alpha(S)$$

$\delta_{I,S}$ tests whether I contradicts any information the system already knows for sure. For example, if the last name has explicitly been confirmed by the user, an information item set I that contains a different last name is inconsistent.

$\delta_{I,I}$ tests whether the currently hypothesized information item set contradicts itself, e.g. if it contains two different last names. (Such hypotheses may well occur if a name can be used as a first name as well as as a last name.)

$\alpha(S)$ is a normalization constant which is independent of I, and can be omitted since it does not contribute to the maximization.

Now we apply the usual maximum approximation, i.e. we replace the sum over W by the maximum[2]:

$$\hat{I} \quad \approx \quad \arg\max_I \left\{ \max_W p(O|W) \cdot \sum_G P(WIG) \cdot \delta_{I,S} \; \delta_{I,I} \right\}$$

3.2 Modelling the Goal's Distribution

We decompose the goal G into two parts (subsets) which have to be evaluated differently: the *language-model goal* G_{LM} and the *database goal* G_{DB}. They are disjoint, and their priors are assumed to be independent, i.e.

$$P(G) \quad = \quad P(G_{LM}) \cdot P(G_{DB}) \tag{2}$$

With this, we obtain

$$\begin{aligned} &\sum_G P(WIG) = \\ &\quad = \sum_G P(WI|G)\, P(G) \\ &\quad = \sum_{G_{DB}} \sum_{G_{LM}} P(WI|G_{LM}G_{DB})\, P(G_{LM})\, P(G_{DB}) \end{aligned}$$

The language-model goal G_{LM} is the part of the goal for which the prior $P(G_{LM})$ is not explicitly available but implicitly modelled by the language model. If, for example, the goal G was to talk to a certain person, G_{LM} would be the request for call completion and G_{DB} the desired person. We get:

$$\begin{aligned} \sum_G P(WIG) \quad &= \quad \sum_{G_{DB}} \sum_{G_{LM}} P(WIG_{LM}|G_{DB}) \cdot P(G_{DB}) \\ &= \quad \sum_{G_{DB}} P(WI|G_{DB}) \cdot P(G_{DB}) \end{aligned}$$

On the other hand, for the database goal G_{DB}, the *a-priori* distribution $P(G_{DB})$ is explicitly available. $P(G_{DB})$ reflects how often a person is asked for and is 0 if the current hypothesis for G refers to a non-existing database entry.

However, we still need a language model that models the sentence *structure*. For example, it does *not* model *whom* to talk to, but *how* the person is specified: by first and last name, by title and last name, by first name only, etc.

This calls for a language model that provides prior probabilities for word-sequence and information-item *templates*, in which all G_{DB}-related words are replaced by a placeholder as in `"Ah, yes, I would like to talk to <first-name> <last-name>, please"`. The word sequence template for W will be called W^T, and the corresponding information item template for I will be named I^T. Thus,

$$\begin{aligned} &\sum_G P(WIG) \\ &\quad = \sum_{G_{DB}} P(WIW^TI^T|G_{DB}) \cdot P(G_{DB}) \\ &\quad = \sum_{G_{DB}} P(WI|W^TI^TG_{DB}) \cdot P(W^TI^T|G_{DB}) \cdot P(G_{DB}) \end{aligned}$$

Since (W, I) is uniquely determined by (W^T, I^T, G_{DB}), $P(WI|W^TI^TG_{DB})$ is either 1 or 0 depending on whether filling in the templates in W^T and I^T from the hypothesized G_{DB} leads to (W, I) or not. Using our matching operator, we abbreviate $P(WI|W^TI^TG_{DB})$ as $\delta_{WI,G_{DB}}$. Furthermore, $P(W^TI^T|G_{DB})$ can be assumed independent of G_{DB} since – at least in our system – all G_{DB} are of the same type. We finally obtain:

$$\begin{aligned} &\sum_G P(WIG) \\ &\quad = \quad P(W^TI^T) \cdot \sum_{G_{DB}} \delta_{WI,G_{DB}} \cdot P(G_{DB}), \end{aligned}$$

where $P(W^TI^T)$ is the template language model. It implicitly models the prior distribution of the language-model goal G_{LM}. For example, to compute the prior probability for the event that a caller wants an e-mail address, we have to sum up $P(W^TI^T)$ for all possible word sequences and corresponding information item sets that contain the request "e-mail address". The structure of $P(W^TI^T)$ is closely related to the speech understanding component and will be discussed in more detail in section 5.

3.3 Decision Rule

We obtain the final decision rule:

$$\hat{I} \approx \arg\max_{I} \left\{ \max_{W} \underbrace{p(O|W)}_{\text{acoustics}} \cdot \underbrace{P(W^T I^T)}_{\text{grammar}} \cdot \underbrace{\sum_{G_{DB}} \delta_{WI,G_{DB}} \cdot P(G_{DB}) \cdot}_{\text{database knowledge}} \underbrace{\delta_{I,S}\ \delta_{I,I}}_{\substack{\text{consistency}\\ \text{constraints}}} \right\} \quad (3)$$

The rule contains long-span dependencies that make a direct evaluation (e.g. by an integrated Viterbi beam search) prohibitive. To overcome this practical problem, we chose a multi-stage approach for our real-time implementation. The incoming speech is processed by a pipeline of processing stages, in which each stage acts as a filter to reduce the search space for the subsequent stage while applying as much knowledge as feasible at that point. The further the search space gets reduced, the more complex knowledge sources can be incorporated. This means in particular, the output of every stage except the last must be a set of alternative hypotheses rather than a single one.

4 Speech Recognition

The first module in the signal processing pipeline is a speaker-independent continuous-speech recognizer. It is a state-of-the-art HMM recognizer that processes spontaneous fluently-spoken natural-language utterances over the telephone in real-time. In terms of our decision criterion, the recognizer provides the acoustic likelihood $p(O|W)$.

The recognizer consumes the very most part of CPU processing time. The computational effort of all other components is negligible in comparison.

4.1 Acoustic-Phonetic Modelling

The recognizer uses 4461 strongly-tied triphones that share 703 tied states. The state emission probabilities are modelled as mixtures of Gaussian densities. The feature vector consists of 12 cepstrum coefficients plus deltas, derived from a Mel-spaced telephone-bandwidth filterbank (MFCC).

The acoustic models have been trained on a large German spontaneous-speech database that we have recorded over the telephone during the field test of our train schedule information system. This corpus comprises 33081 utterances, which amount to a total of about 12 hours of actual speech.

We use a rule-based state-tying approach based on the concept of diphone-like units [13]. For our German task, this performs comparably to what we achieved with data-driven bottom-up tying [21, 5]. In addition, it allowed us to synthesize 958 of the 1985 triphones in our (PADIS) recognition lexicon that have not been observed in the (train schedule) training corpus.

4.2 Word-Graph Interface

Rather than outputting a single best sentence, the recognizer creates a set of plausible sentence hypotheses. The final decision is deferred to the subsequent language understanding module, which incorporates the additional knowledge sources.

The sentence alternatives are represented as a compact word graph [14]. A word graph (see figure 4) 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. The sum of all scores on a path is the total negative log-likelihood of the respective word sequence, $-\log p(O|W)$.

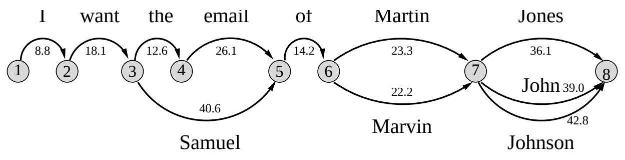

Figure 4: Example word graph.

We use the *time-conditioned tree copies* method to create the word graph [14]. In this algorithm, word-graph generation is implemented as a two stage process. The first stage, the word-hypotheses generator, performs a time-synchronous Viterbi beam search with a tree-organized lexicon. A search tree is started for every time frame. For every frame, all ending words are collected and transferred to the second stage, the word-graph optimizer. It combines these word hypotheses into a pruned word graph.

We have also experimented with another method to create word graphs that uses *word-conditioned tree copies* as described in [11]. The CPU effort for both methods turned out to be nearly the same in our task, which is in line with the results in [15].

[2]In the practical implementation, the maximization over W is restricted to word sequences whose interpretation is I (for all other W, $P(WIG)$ is zero).

4.3 Language Model

Although the recognizer only has the task to provide word hypotheses rather than to decide on the best path, it is still crucial to use a language model to properly focus the beam search. A language model is also used to prune the word graphs in order to keep the word-graph density low. The word-graph density has a direct impact on the computational requirement of the subsequent speech understanding stage.

We use a bigram model that is independent of dialogue history and database. All database terminal symbols (words directly referring to a database entry, such as last names) are modelled by word categories with flat distributions within each category [12].

4.4 Vocabulary Generation

The recognition lexicon is created by an automatic procedure. First, a word list is composed of:

- the database terminals (e.g. all last and first names, titles, etc. occuring in the database);
- the terminal symbols of our speech-understanding grammar (about 140);
- an additional manually-maintained list of about 270 words including popular out-of-vocabulary words that could cause misrecognitions, special words for non-speech sounds (like hesitations), and often-used pronunciation variants of terminals.

Then, for every word in the word list, a phonetic transcription is generated automatically using large background lexica. Words whose transcriptions cannot be found in these lexica are transcribed using a statistical method [4]. For foreign names that are regularly mispronounced or cannot be transcribed automatically, we permit pronunciation variants to be directly stored in the telephone directory database.

During vocabulary generation, database terminals are tagged in the recognition lexicon. E.g., the lexicon entry for the first name Sally is `Sally:fname`. If a name can be a first as well as a last name, two differently tagged homophones will be generated. The advantage is that the subsequent speech-understanding stage's parser can identify database terminals by the tag without referring to the database or having to search large terminal lists.

5 Language Understanding

The language understanding component computes the information item set I from an utterance's word graph. It actually has two tasks:

- to find the most probable path through the graph, and
- to compute its meaning.

For finding the best path, language understanding contributes the template language model $P(W^T I^T)$, the consistency tests $\delta_{I,S}$ and $\delta_{I,I}$, and the integration of the database knowledge.

For computing the meaning, language understanding contributes a parser. Since we have to deal with spontaneous natural language, we cannot expect the input to be grammatically correct. Recognition errors may also account for input hypotheses that are not completely parsable. Therefore, a very robust parser which can handle erroneous or grammatically incorrect input is needed.

These two tasks are in principle conceptionally independent, but practically dependent on each other: First, we take over the internal structure of the template language model from the parsing grammar, and second, consistency tests and database integration are performed on the meaning level.

5.1 The Concept Parser

Since we are only interested in an utterance's semantic information I, the system does not have to understand the semantics of every word in the input, but only those phrases which contain task-relevant information. These meaningful phrases (we call them *concepts*) are usually well structured and can be modelled by a simple context-free grammar [16, 8, 3]. Using a partial-parsing strategy, the concepts can be identified in the input whereas meaningless sequences (fillers) do not affect the parse.

The individual concepts are modelled by an attributed stochastic context-free grammar with a distinct start symbol for each concept. A top-down chart parser is used to search the word graph for meaningful phrases. The result of this parse is stored in a *concept graph*. It has the same nodes as the underlying word graph, and its arcs are concept instances found by the parser. Every word sequence that can be parsed as a concept leads to a corresponding concept arc. Figure 5 shows the concept graph that was created from the word graph in figure 4.

Unparsable parts of the utterance lead to gaps in the concept graph. This would make it impossible to extract

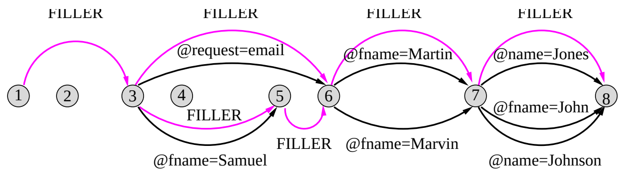

Figure 5: Example concept graph.

the best path through the concept graph. Therefore, these gaps are bridged by so-called *filler arcs*. In addition, a filler arc is added in parallel to every concept arc as an alternative interpretation, since a concept may have been found only due to a recognition error.

Filler-word sequences actually serve a twofold purpose: First, they allow avoiding complicated grammatical constructions as e.g. *"Please, would you be so kind to connect me to Mr. Jones,"* where it is sufficient to spot the phrases *connect* and *Mr. Jones*. Second, they do some out-of-domain and out-of-vocabulary modelling by e.g. covering phrases like *"Connect me to Burger King."*

5.2 The Template Language Model

The language understanding component provides the template language model $P(W^T I^T)$. We regard W^T and I^T as a joint event to account for the variety of ways a certain information item set can be expressed on the one hand and the fact that multiple interpretations of a word sequence may exist on the other hand – although this rarely happens in our task. If I^T is not an interpretation of W^T, $P(W^T I^T)$ is simply 0.

$P(W^T I^T)$ has a two-layer structure as described in [1]. As a result from the parsing process, every path through the word graph becomes segmented into a sequence of concepts and fillers. With C denoting such a concept/filler segmentation, we can rewrite $P(W^T I^T)$ as:

$$P(W^T I^T) \quad = \quad \sum_C P(W^T I^T|C)P(C)$$

We decompose $P(C) = P(c_0, c_1, ..., c_{N+1})$ into a bigram model, and we assume that the conditional probability for a c_i's partial word sequence $(W^T_{c_i}, I^T_{c_i})$ only depends on c_i:

$$\begin{aligned} P(W^T I^T) \quad &= \quad P(c_{N+1}|c_N) \\ &\cdot \prod_{i=1}^{N} P(W^T_{c_i} I^T_{c_i}|c_i)P(c_i|c_{i-1}) \end{aligned}$$

where c_0 and c_{N+1} refer to the sentence start and end, respectively. Using an N-gram model to model the sequence of concepts is in line with most traditional systems that use context-free grammars or finite-networks for partial parsing, (cf. [10, 16, 7]).

For regular concepts, $P(W^T_{c_i} I^T_{c_i}|c_i)$ is provided by the the rule probabilities of the concept grammar: Every rule is assigned a probability that indicates how likely it is to be applied given the left-hand side non-terminal. The probability of a concept derivation is equal to the product of the production probabilities of all rules applied to parse this concept (cf. [6]).

For filler arcs, the language model score is provided by a stochastic word-level bigram language model. Since the fillers model those parts of the word graph outside the concept grammar, we trained the filler bigram on those parts of the training corpus that were not covered by the grammar.

The template language model is implemented as follows: During the parse, each concept arc is assigned a score which consists of the acoustic score of the concept's word sequences plus the grammar score. When the filler arcs are added to the concept graph, each filler arc is assigned the score of the best word sequence from its start to its end node (including the bigram score). The concept bigram is applied later when we extract paths from the concept graph.

5.3 Computing the Meaning

At the same time that the input is parsed top-down, the meaning is computed in a bottom-up manner: Every non-terminal can be assigned a set of *attributes*. Their values are computed when the non-terminal is expanded using a (syntactic) grammar rule. For each syntactic rule, there may be semantic rules which determine how the values of the attributes are computed from the attributes of the non-terminals used on the right-hand side of the syntactic rule.

Figure 6 shows example rules of our grammar. The values in parentheses are observation counts from the training corpus. The rule probabilities are estimated on these counts.

```
<affiliation> ::= (61) in the <db_group> <dept>
        group := <db_group>.group

<affiliation> ::= (83) of the <db_group> <dept>
        group := <db_group>.group

<dept>        ::= (127) group
<dept>        ::= (32) department


  // import group names from database
<db_group>    ::= DATABASE("group") //terminals for 'group'
        group :=  *   //use terminal as attribute value
```

Figure 6: Example grammar rules.

The word lists and attributes 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. Actually, we do not have to consult the database to resolve these non-terminals, because database terminals have been tagged during the automatic vocabulary generation process (cf. section 4.4). The attribute value is directly derived from the word itself.

5.4 Integrating the Database

As the last step to make the final decision for one sentence hypothesis, the best path through the concept graph has to be computed. If we had no further knowledge sources, we could use a standard Viterbi algorithm to extract the best concept sequence from the concept graph. However, in order to incorporate the database knowledge (the goal's prior distribution $P(G_{DB})$) and the consistency constraints $\delta_{I,S}$ (dialogue history) and $\delta_{I,I}$ (consistency within the interpretation itself), a more complex approach has to be taken:

We use an N-best approach in which N, the number of paths that have to be computed, does not have to be known in advance, but the sentences hypotheses are computed one after the other, sorted by their concept-graph scores.

We start by extracting the first best path and update its path score using the full model. Then we extract the next best and rescore it, too. This is repeated over again until we heuristically decide that the best sentence hypothesis (the one with the best score according to the full model) is expected to be included in the N best hypotheses we have examined. To ensure real-time processing, we must limit N to $N_{\max} \approx 50$. If no consistent path was found amongst the top $N_{\max}$ hypotheses, we set $\hat{I} = \emptyset$.

Consistency with the Database

Consistency with the database is ensured by the *a-priori* probability $P(G_{DB})$: For an information item set I of a sentence hypothesis that does not refer to at least one existing database entry, $P(G_{DB})$ will be 0, and therefore the hypothesis can be rejected immediately.

The following example[3] illustrates this. For all examples, we assume that 'Martin Jones' is a valid database entry, whereas 'Marvin Jones' and 'Martin Johnson' are not.

Best path:
```
I want the email of Marvin:fname Jones:name
```
db query: last name: Jones
first name: Marvin
$\Rightarrow$ No database match: hypothesis gets rejected.

2nd best path:
```
I want the email of Martin:fname Jones:name
```
db query: last name: Jones
first name: Martin
$\Rightarrow$ Consistent.

[3]The examples have been translated from original German dialogues collected in the PADIS field test. Only the personal data (e.g. the names) have been changed.

Matching Rules

The consistency constraints $\delta_{I,S}$ and $\delta_{I,I}$ are implemented by *matching rules*. In the following, we explain the rules actually employed in our prototype.

Rule 1: Consistency Within Interpretation Itself A hypothesized interpretation of the user's response is only accepted if it does not contain contradictory values for an attribute ($\delta_{I,I} \neq 0$). Example:
```
I want the email of Martin:fname John:fname
```
$\Rightarrow$ Two different values for the same item.

Rule 2: Match With System Prompt According to $\delta_{I,S}$, an interpretation that contains a correction of some item not occurring in the system's prompt is rejected. Example:

System: `Thus, you would like to talk to Mr. Jones?`

Best path:
```
no not Johnson:name
```
$\Rightarrow$ Inconsistent with the system prompt.
...
4th best path:
```
no not Jones:name
```
$\Rightarrow$ Consistent and selected.

Rule 3: Match With System's Belief Interpretations which, after combination with the system's belief, do not refer to a valid database entry are also rejected according to $\delta_{I,S}$. For example:

System: `Which Martin would you like to talk to?`

Best path:
```
Doctor:title Johnson:name please
```
db query: last name: Johnson
first name: Martin
title: Doctor
$\Rightarrow$ No database match.
2nd best path:
```
Doctor:title Jones:name please
```
db query: last name: Jones
first name: Martin
title: Doctor
$\Rightarrow$ Consistent and selected.

N-best Algorithm

The procedure described above requires the N best concept sequences to be extracted from the concept graph, where N is not known in advance. A number of algorithms have been developed for exact or approximate iterative generation of N-best hypotheses [19, 17]. We use the exact algorithm described in [20], which was improved to achieve a computational complexity linear in N.

As a preparation, the algorithm first performs a standard Viterbi optimization, i.e. for every concept arc, the best partial path from the sentence beginning is computed, and its partial score and backpointer are stored along with the arc in the graph data structure. To extract the first best path, one just has to find the best ending arc and follow the backpointer chain. When looking for the next best path, we do basically the same but avoid outputting the first best again. In general, extracting the nth best path while knowing the $n-1$ best hypotheses is seen as extracting the best path while excluding the $n-1$ best paths from search space.

We keep track of the $n-1$ best paths by storing them in a compact *backward tree*. Figure 7 shows an example of a concept graph and the corresponding backward tree after extracting the $n-1=4$ best sentences. When extracting the next best hypothesis, we exploit the fact that it must join the tree at some node (which may be the root node).

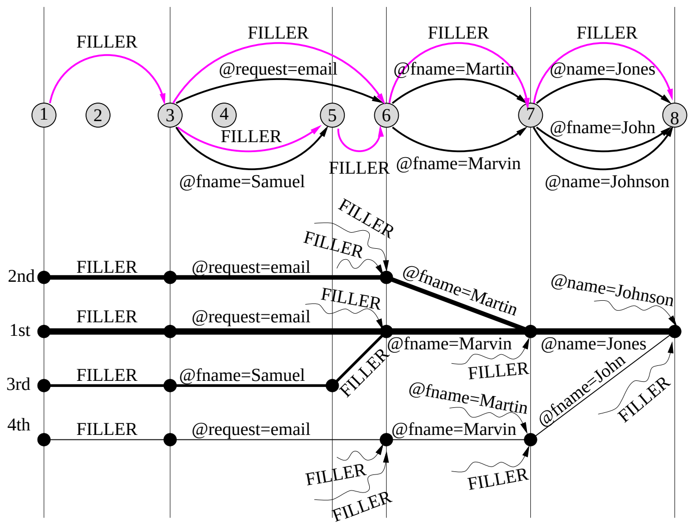

Figure 7: Example of an N-best backward tree. The next best path joins the tree via one of the serpentined arrows. Now select the one that leads to the best total sentence score.

Thus, to find the nth path, we determine at *which* node it joins, and via which graph arc: for each currently existing tree node, we determine the best entering path (i.e. the one leading to the best sentence score), and then select the best one from all nodes. This selection is the nth best hypothesis we are looking for. Finally, we add this path to the backward tree to prepare for the next best.

The best path entering a certain tree node is found by optimizing over all arcs entering the corresponding graph node but skipping all arcs for which already a corresponding tree arc exists, because these tree arcs represent one or more of those $n-1$ best paths that we have to exclude. The tree node and the entering graph arc define a sentence hypothesis:

- The graph arc determines the head of the path, which is the best partial path from sentence beginning. We already know it (and its partial score) from the initial Viterbi optimization – just follow the backpointer chain.
- The tree node determines the tail of the path (by its parent arcs up to the tree root).

With this, computing the score of this whole sentence path becomes inexpensive. It consists of three components, the partial path score of the head, the concept-bigram transition score from the entering arc to the tree node's parent arc, and the partial path score of the tail (computed once for every arc added to the tree).

This optimization has to be performed for every node in the backward tree, which grows with every extracted hypothesis. Thus, we obtain a computational complexity of $O(N^2)$ for the extraction of N best sentences, as described in [20]. However, for most nodes, the optimization result does not change from n to $n+1$; only for tree nodes on the path of the nth sentence hypothesis, the result may differ when extracting the $n+1$st hypothesis. These redundant computations can be avoided by performing this optimization whenever adding a path to the backward tree, i.e. only once for every node on this path. A similar strategy can be followed in selecting the best joining node. The computational effort for extracting the nth best sentence now is in the order of magnitude of extracting the single best sentence from a word graph, and N best hypotheses can now be computed with linear complexity $O(N)$.

6 Dialogue Control

The overall understanding task of the system is to find the caller's dialogue goal G, i.e to find out what person (database entry) he is talking about and what service (e.g. call completion or email address) is requested. For the dialogue control module, this means that a number of information slots in a query pattern have to be filled.

In each dialogue turn, the language understanding component provides a set of information items which were extracted from the user's last utterance. Since a single user utterance usually does not contain all the information necessary to provide the service, the system needs capabilities to ask for missing information. In addition, the user must be able to detect and correct possible understanding errors. This calls for a dialogue between the system and the user.

In PADIS, the system keeps prompting the user for missing information as long as a unique database entry cannot be identified by the information collected from the user's previous utterances. Sometimes however, the user does not know the information he is prompted for, but would be able to give other information that could be used for disambiguation. For example, the user cannot answer the question for a person's first name, but may know the department where this person is working. To handle such cases, we need a flexible mixed-initiative dialogue strategy rather than a rigid system-driven interaction. In our system, the user can at any given time specify more or different information than he is actually asked for.

The disambiguation strategy followed by the system in order to get more specific information on what is requested by the user is termed as *slot filling*. It was already used in our previous train-timetable system [3], and works as follows: The system maintains an ordered list of information slots (last name, first name, group, etc.). It keeps asking questions on the next item of this list as long as further disambiguation is needed. E.g. knowing only the first name elicits a question for the last name, then for the group, and so on. This slot-filling strategy allowing the user to freely specify information items is illustrated in figure 8.

System:	*Hello, this is PADIS. How can I help you?*
Caller:	*I'd like to talk to Mr. Anderson.*
System:	*What is the first name of Mr. Anderson?*
Caller:	*Uh, I don't know — but I think he is working somewhere in the administration.*
System:	*So you want to talk to Mr. Paul Anderson from the administration department?*
Caller:	*Yes, sounds good.*
System:	...

Figure 8: Dialogue example demonstrating the system's slot-filling strategy.

In most cases, the user does not have to fill all slots in the query pattern. He only has to provide enough information to uniquely identify the desired service and database entry. E.g. in some cases a first name may already uniquely specify a person, whereas in other cases more information is needed. To avoid superfluous questions, in order to keep the dialogues as short as possible, the dialogue component consults the telephone database after each user turn to determine whether or not further information has to be requested (cf. the example dialogue in figure 1, where the first name Mike is already unique, whereas Mary needs further disambiguation). This integration of the database into the dialogue control is very efficient since it keeps the dialogues short while complex disambiguation dialogues are still possible.

In an automated system, recognition errors may and do occur. It is therefore important that the user has the possibility to detect and correct these errors before he may be connected to the wrong party. In the PADIS system, the information understood in the previous turn is repeated in the next question (E.g.: `What is the first name of Mr. Jones?`) The user may correct any incorrect information. If no correction is made, the system considers this information to be verified. This method is called *implicit verification*. Only if no further information has to be asked for, a pure confirmation question is asked (*explicit verification*). These correction mechanisms are illustrated in figure 9.

System:	*Hello, this is PADIS. How can I help you?*
Caller:	*Mary White.*
System:	*What is the first name of Mr. Wright?*
Caller:	*No, I don't want Mr. Wright, I would like to talk to* `Mrs. Mary White.`
System:	*So you want to talk to Mrs. Mary White?*
Caller:	*Yes, you got it.*
System:	*I connect you ...*

Figure 9: Dialogue example demonstrating the system's verification strategies: a) In turn 2, the wrongly recognized *Mr. Wright* is stated for implicit verification and is corrected by the user. b) In turn 3, the system explicitly verifies *Mrs. Mary White* which is correct now.

Dialogue modelling in the PADIS application may attain a considerable complexity. This is due to the following facts: a) The user may specify the information slots in an arbitrary order. b) An arbitrary number of slots may be specified. c) The possible correction of misrecognized slots by the user adds further complexity.

Designing the PADIS dialogue flow by means of graphs or finite-state automata would result in impracticably large networks and may even be infeasible for more complex applications. Therefore, a special dialogue description language was developed [2], in which the task-specific aspects like slot definitions, questions, and verification phrases can be specified in a declarative way. The dialogue module itself incorporates the general knowledge on how to conduct information-inquiry dialogues as the slot-filling mechanism described above or the automatic generation of verification questions [2]. This combination of the built-in general dialogue strategy and a programming language to describe the task specifics reduces the complexity of a dialogue description and therefore makes it possible to extend the functionality of an application or to create a new application quite easily.

7 Speech Output

In PADIS, communication between the user and the system is a two-way process which requires the system to talk to the user and not only vice versa. The dialogue description provides templates for phrases which can be used to create system prompts. Those templates determine, for example, how a certain slot is asked for, and how its content is formulated in implicit or explicit verifications.

In the information gathering dialogue, several of these templates may be concatenated and filled with values from the system's current belief. In some cases, 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 talk to White?", we use "to Mrs. White" instead, even if the user didn't say "Mrs.". This sounds more natural to the caller. However, this feature has to be used with care: We observed that users easily get confused if the system comes up with something the user has not said, especially if it is wrong due to recognition errors.

Specific templates also exist for the output of directory information. They are filled with values from the respective listing. In addition, the dialogue description provides phrases for greeting and meta-dialogue (e.g. "Do you want any further information?").

For the output of the prompts, PADIS concatenates prerecorded phrases. This sounds more natural than today's state-of-the-art speech synthesis systems and is still feasible for a relatively small database. For large and dynamically changing databases however, this approach is not practical any more, because for every new entry in the database the corresponding phrases must be recorded.

8 Field Test

The PADIS system has been used in a field test in our lab since March 1996. About 7000 dialogues were collected between March and December 1996. Of these, about 6000 were transcribed and evaluated, corresponding to approximately 7 hours of user speech. Table 2 summarizes the most recent PADIS corpus.

Table 2: Characteristics of latest corpus.

no. of dialogues	6124
no. of turns	18112 (3.0 per dialogue)
no. of words	45987 (2.5 per turn)
time estimate	7h
no. of out-of-vocab. words	822 (1.8%)

8.1 Task Characteristics

The PADIS database contains about 600 subscriber entries. A user may request phone, fax, email, and work place location information on the one hand, and direct call completion on the other hand.

The total vocabulary consists of 1526 words, most of which are database terminal symbols, i.e. words that refer to database entries (see section 4.4). Table 3 shows an overview of the PADIS vocabulary.

Table 3: Composition of vocabulary.

word category	# words	percentage
last names	687	
first names	304	
genders	2	
groups	121	
sites	4	
titles	2	
$\sum$ database terminals	1120	73%
grammar words	139	9%
filler words	267	18%
total	1526	100%

8.2 System Performance

In the current system, more than 95% of the dialogues are successful, i.e. in more than 95% of the calls, the users finally get the service they request.

We made an offline evaluation of the system performance for three setups:

- First, we used a traditional system without consistency checks (a simple first-best system, no N-best rescoring).
- In the next step, the first matching rule (consistency within utterance) and database knowledge were used.
- Finally, all matching rules described above were applied.

The results given in this section were evaluated on an earlier corpus (3512 dialogues), which was split into a training and a test corpus as shown table 4. The acoustic models have been trained on 12.1h of speech data from our train schedule information system.

Table 5 shows the word error rate WER, the average search depth $\bar{n}$, the test-set perplexity PP, and detailed information on the concept error rate CER for the test

Table 4: PADIS field-test corpus characteristics.

	Training	Test
dialogues	2348(4.5h)	1164(1.5h)
turns	8214(3.5 per dial.)	3198(2.7 per dial.)
words	26445(3.2 per turn)	8743(2.7 per turn)

corpus. The concept error rate measures insertions, deletions, and substitutions of the (attributed) concept sequence. The number of insertions, deletions and substitutions for each setup is also shown. To show the effect of the database integration, the substitutions were split into attribute substitutions S_{Attr} (the correct concept was understood, but its attributes were wrong) and concept substitutions S_{Conc} (a wrong concept was understood). The average search depth $\bar{n}$ is the rank of the selected hypothesis in the N-best search and indicates how many hypotheses were rejected until a consistent path through the concept graph was found. The maximum search depth $N_{\max}$ in these experiments was 100.

In interpreting the error rates, it should be noted that a dialogue can be successful even if some attributes are misrecognized, e.g. a deletion of a first name when the last name is unique.

By applying the within-turn constraints, a relative reduction by 17% in concept error rate, 15% in WER, and 35% in perplexity has been achieved. Attribute substitutions have been reduced by 36%. Since the largest part of the error reduction is due to rejecting hypotheses referring to non-existing database entries, the gain perceived by the users is much greater: Users turned out to be especially annoyed if a non-existing person is understood, because the computer "could have known".

The additional use of the system's belief and the current question has only led to a slight improvement on CER and has had nearly no effect on the WER. This is because the effects of the dialogue history inevitably remain small in dialogues consisting of just three turns on average. The shortest possible dialogue already takes two turns: "*How can I help you?* – Mr. Jones, please. – *You want to talk to Mr. Jones?* - Yes."

8.3 User Behaviour

Who is using the system?

If one wants to judge the usability of an automatic inquiry system, it is important to look at the people who use the system. In our case, the callers are mainly employees of the Philips research laboratories in Aachen/Germany. Most of these users have academic degrees, most of them with a technical background. Even though many of the users of PADIS do not work in speech processing but in fields like lighting or material science, many of them have already used our automatic train timetable information system [3].

How is the system used?

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 manual dialing.

We examined the way that users talk to our system for about 6000 dialogues. In particular, we concentrated on the caller's first utterance in each dialogue (immediately after the system's greeting) in which we assumed the caller spoke according to his own preferences. In 57% of the calls, the user gave the first and the last name of the desired party, while in 36% (5%), the caller specified the last name (first name) only.

One might expect humans talking to a machine to use the simplest possible formulation, which would be first name or form of address followed by the last name for connection requests[4]. Interestingly however, we have found this formulation in only 32% of the first dialogue utterances. Most users preferred much more talkative formulations: The average length of the first dialogue utterances is 3.8 words. In over 30%, "*please*" was used. 28% of the utterances contain phrases like *"I'd like to"*.

Although these figures might be somewhat biased due to our caller sample, it seems that if users are not restricted by the system, they prefer to talk the same way they would talk to a human, despite the extra verbiage. However, if we compare these results to the numbers derived on an earlier corpus [9], we find that the users were even more talkative when the field test started. This might indicate the trend that users are going to develop a new concise style to give their commands to the machine.

System prompts and dialogue efficiency

The duration or conciseness of the system prompts elicited the widest range of disagreement among users. 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 replacing 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,

[4]If the caller does not ask for a particular service, the system assumes that a connection is requested.

Table 5: Results on the PADIS field-test corpus.

Setup	PP	$\bar{n}$	WER	Sub_{Attr}	Sub_{Conc}	Ins	Del	CER
First best	25.1	1.0	28.9%	768	441	271	527	33.4%
Consistency within turn	16.2	3.2	24.6%	487	429	231	541	27.8%
Full model	n/a	3.8	24.4%	476	375	240	544	26.9%
Relative Gain	(35%)		15%	38%	15%	11%	-3%	19%

the system does not output the number of the desired party when transferring a call. The user is only given the telephone number if the line is busy.

Another possibility of further improving dialogue efficiency would be to get rid of the need to verify every information item. We have been investigating confidence measures for information items to be able to suppress verification questions for reliably recognized items. However, this technique was not used in the experiment presented here.

Randomized prompts

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.

How to handle inconsistent information?

As mentioned earlier, users find inconsistent information as in *Mrs. Mike White* unacceptable. In general, the integration of database constraints avoids this problem. However, 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 selected 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.

9 Conclusions

In this paper we have presented the Philips automatic directory information and exchange board system PADIS. We have derived a stochastic framework for speech recognition and language understanding which incorporates database knowledge and dialogue history. The resulting decision rule is implemented in a multi-stage process, where in each stage, the search space is reduced while additional knowledge sources are employed.

The new decision rule improved the concept error rate, which measures understanding accuracy, by 19%. In particular, attribute substitution errors, which are the most annoying for the user, was reduced by 38%. We believe that using a method of this type will become even more essential if a much larger database is used.

The mixed-initiative dialogue of PADIS, which also integrates the database in order to keep the dialogues as short as possible, enables the user to access an entry in the telephone database even though he may not know some of the information he is asked for.

We have reported results on a field test of the system which has been conducted in our research lab. Our data suggest that users like the opportunity to speak naturally to our automatic system. Despite the difficulties inherent in understanding unrestricted natural language, our system achieved a dialogue success rate of about 95%.

References

[1] H. Aust and M. Oerder. A real-time prototype of an automatic inquiry system. In *International Conference on Spoken Language Processing*, volume 2, pages 703–706, Sep. 1994.

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

[3] H. Aust, M. Oerder, F. Seide, and V Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 17(3–4):249–262, Nov. 1995.

[4] S. Besling. A statistical system for grapheme–to–phoneme conversion. In *Proc. Tenth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research: Reflections on the Future of Text*, pages 5–13, Waterloo, Canada, Oct. 1994.

[5] C. Dugast, P. Beyerlein, and R. Haeb-Umbach. Application of clustering techniques to mixture density modelling for continous–speech recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 524–527, Detroit, MI, May. 1995.

[6] K.S. Fu. *Syntactic Pattern Recognition and Applications*, pages 196–245. Prentice-Hall,1982.

[7] E.P. Giachin. Phrase bigrams for continuous speech recognition. In *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Proc.*, pages 225–228, Detroit, 1995.

[8] F. Jelinek, J. Lafferty, and R. Mercer. Basic methods of probabilistic context free grammars. In *NATO ASI*, pages 345–360, Cetraro, Jul. 1990.

[9] A. Kellner, B. Rueber, and F. Seide. A voice-controlled automatic telephone switchboard and directory information system. In *IVTTA*, pages 117–120, Basking Ridge, NJ, Sep. 1996.

[10] M. Meteer and J. R. Rohlicek. Statistical language modelling combining N-gram and context-free grammars. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume II, pages 37–40, Minneapolis, April 1993.

[11] H. Ney and X. Aubert. A word graph algorithm for large vocabulary, continuous speech recognition. In *International Conference on Spoken Language Processing*, volume 4, pages 1355–1358, Yokohamai, Japan, Sep. 1994.

[12] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[13] H. Ney, D. Mergel, and S. Marcus. On the automatic training of phonetic units for word recognition. *IEEE Trans. Acoust. Speech and Signal Processing*, 34(1):209–213, 1986.

[14] M. Oerder and H. Ney. Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume II, pages 119–122, Minneapolis, April 1993.

[15] S. Ortmanns, H. Ney, F. Seide, and Lindam I. A comparison of time conditioned and word conditioned search techniques for large vocabulary speech recognition. In *International Conference on Spoken Language Processing*, volume 4, pages 2091–2094, Philadelphia, PA, Oct. 1996.

[16] R. Pieraccini and E. Levin. Stochastic representation of semantic structure for speech understanding. In *Proc. of the EUROSPEECH*, pages 383–386, Genua, 1991.

[17] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 701–704, May 1991.

[18] F. Seide, B. Rueber, and A. Kellner. Improving speech understanding by incorporating database constraints and dialogue history. In *International Conference on Spoken Language Processing*, volume 2, pages 1017–1020, Philadelphia, PA, Oct. 1996.

[19] F. K. Soong and E.-F. Huang. A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 705–708, Toronto, Canada, May 1991.

[20] B.H. Tran, F. Seide, and V. Steinbiss. A word graph based N-best search in continuous speech recognition. In *International Conference on Spoken Language Processing*, volume 4, pages 2127–2130, Philadelphia, PA, Oct. 1996.

[21] S. J. Young and P. C. Woodland. The use of state tying in continuous speech recognition. In *Proc. EUROSPEECH*, pages 2203–2206, Berlin, Germany, Sep. 1993.