Studies on Speech Recognition based on Discriminative Statistical Models and Heuristic Search Strategies

Tatsuya KAWAHARA

December 1994

Abstract

Automatic recognition of conversational speech involves comprehensive studies of pattern recognition, knowledge representation and search algorithms. It is significant for intelligent and friendly human-machine interfaces. In a spoken language understanding system or a dialogue system, it is essential to cope with a variety of speakers and natural continuous utterances including ill-formed ones.

In order to realize highly accurate and robust recognition of conversational speech, in this thesis, we present discriminative statistical models for the phone recognizer, and address heuristic search strategies to combine the acoustic model and the language model. We adopt the following approaches. At first, discriminant analysis is incorporated to Hidden Markov Model (HMM) to improve both discriminative ability and robustness. Next, admissible and efficient search algorithms are explored with simple linguistic knowledge as heuristics. In the framework of A* search, statistical knowledge sources such as syllable or word bigram and symbolic knowledge sources such as syntax and semantics are integrated into a single evaluation function, of which heuristic score is computed by the statistical ones. Moreover, we propose a model of interaction between the speech recognizer and the natural language analyzer that are conventionally cascaded. The models and the algorithms are implemented for both parsing and spotting. The parsing approach searches for an optimal word sequence under a grammatical constraint on complete sentences, while the spotting approach extracts keywords or key-phrases necessary for understanding, skipping the unrecognizable parts.

In Chapter 3, we present discriminative statistical models for the acoustic model, which is the basis of the whole recognition system.

We propose a new model named Pair-Wise Discriminant HMM (PWD-HMM) that couples discriminant analysis and HMM. In the conventional algorithms, phone HMMs are constructed and trained independently using in-class data only, and no considerations are taken for separating different classes. In our approach, for every pair of the classes (observation symbols / HMM states), optimal feature extraction is performed based on discriminant analysis to construct a Bayes classifier. Each classifier ranks the two classes and computes a relative value of the probabilities. In continuous HMM, output probabilities of the states are obtained by combining and normalizing the results of the pair-wise classifications. In discrete HMM, the frame-wise symbols are determined by combining the pair-wise results.

We first apply pair-wise discrimination method to recognition of static segments of speech. Then, discrete HMM and continuous HMM based on pair-wise discriminant analyses are implemented and compared. They achieved a phoneme recognition rate of 86.3%, and a 653 word recognition rate of 86.0%, which are higher by 10% than the conventional HMMs.

In Chapter 4, we discuss heuristic search strategies for continuous speech parsing with the phone HMMs front-end.

We first address A* search algorithms for an LR (context-free) parsing. The widely-used beam search evaluates the hypotheses based on the searched part, and may prune the optimal one on its way. Here, we realize A* search that evaluates each sentence hypothesis with the matching score plus the heuristic score of the unsearched part. For heuristics computation, we propose to use word-pair constraint, which satisfies A*-admissibility and approximates the grammar well with economical complexity. Compared with beam search, the algorithm achieved a better sentence recognition rate (63.0%) with less computation.

In this framework, we integrate various linguistic and pragmatic knowledge sources. First, both the grammar and the heuristic constraint are modified to reflect statistics. A probabilistic LR grammar and word bigram take the place of the deterministic grammar and the word-pair, respectively. Next, the dialogue-level knowledge source is incorporated to predict the possible syntax of user utterances and reduce its perplexity. Then, we make an interaction of the speech parser and the semantic analyzer, which realizes incremental semantic analysis and interactive semantic verification of the partial sentence hypotheses. By integrating all the available knowledge sources, the semantic accuracy was improved by 5$\sim$10% and reached 75.5% in the best case.

In Chapter 5, we address word and phrase spotting algorithms using a heuristic language model.

The conventional spotting methods based on length-free bottom-up matching are sensitive to the local noise or similarity and generate a lot of unnatural false alarms. We propose an evaluation function that reflects not only the matching score of the word itself but also the plausibility of the rest part as a sentence. For the heuristic evaluation for the latter, several language models such as syllable bigram or word-concatenation are compared. The algorithm obtained a much higher detection rate at the 219 vocabulary spotting experiment.

We further propose to adopt a phrase as the spotting unit. Since its syntax is rarely violated and it makes a semantic case, the unit of a phrase is suitable for robust understanding of ill-formed utterances. By choosing the heuristic language constraint so as to be subset of the phrase syntax, the phrase spotting algorithm can be implemented as A* search. It improved the detection rate, compared with the normal word spotting. We also discuss the search strategies for sentence understanding based on the phrase spotting. With an optimal search to find the best hypothesis that satisfies both the phrase syntax and the inter-phrase semantic constraint, the spotting-based approach was comparable to the continuous speech parsing for grammatical utterances, and superior for ill-formed ones.

The components mentioned above are combined into a speech recognition system. It first applies a simple statistical language model, which roughly decodes the input speech and will provide heuristics for the following processing. The efficient parsing is performed to find an optimal sentence hypothesis supposing a grammatical utterance. If it fails, then the spotting approach is taken to cope with ill-formedness. Thus, the system works well for both grammatical and ill-formed utterances. It is integrated with a natural language processor and a task manager into a spoken dialogue system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spoken language is one of the most natural way of communication for us. It was used even before the characters were invented, and babies learn to speak before reading and writing. Even in today's modern society, telephone and television are more popular media than postal mails and books.

However, present communication through or with computers is mainly based on written language. This constraint on media makes it difficult to automate a lot of transactions such as information retrieval, guidance and booking. Furthermore, it also hampers realization of user-friendly interfaces just like human-human communication. Therefore, it is desirable to incorporate spoken language as medium for human-machine interfaces.

Studies on automatic speech recognition (ASR) started about several decades ago[1],[2], and have been active, involving signal processing, pattern processing, language processing and knowledge processing along with statistics, phonetics, psychology and medicine.

The most fruitful achievements in the 1970s were speech analysis based on linear predictive coding (LPC)[3] and pattern matching with dynamic programming (DP)[4]. As a result, recognition of dozens of words became practical, and several spoken language recognition systems were built[5],[6],[7],[8]. However, all of them were speaker-dependent systems that must be trained for every speaker. They required a user to utter several hundreds of speech samples beforehand. Moreover, most of the systems assumed that every word be isolated, which means they required a user to put a pause between words. These constraints hamper user-friendliness greatly and are not realistic in the practical stage.

In the 1980s, speaker-independent continuous speech recognition has been vigorously studied. Among several approaches, the most successful and still promising is the stochastic approach. As an acoustic model, hidden Markov model (HMM) was introduced[9] and demonstrated as powerful in capturing variation and time-warping of speech[10],[11]. As a language model, stochastic models like bigram or trigram (N-gram) got widely used because of its simple and all-embedding structure[12]. These models together have been shown as effective and contributed to speaker-independent speech recognition systems[13].

As continuous speech recognition was put into an experimental stage, it got widely recognized that extracting the meanings of utterances is more significant than transforming them into sentences. Namely, the goal should be understanding, rather than dictating. It is quite natural when we consider the applications of speech technology. One of the ultimate human-machine interfaces is a realization of dialogue with a user. In this case, the system has to deal with natural conversational speech, which demands robustness as well as high accuracy. The conventional modelings mentioned above are insufficient and lack robustness for this purpose.

The problems arise from the following reasons.

- As acoustic modeling, HMM lacks viewpoint of discriminating confusing classes, which is the basis of pattern recognition.

- The conventional language models or parsing approach cannot cope with ill-formed utterances that often happen in real dialogue.

- To deal with tasks of large perplexity and ill-formedness, tight integration of a speech recognizer and a natural language analyzer is necessary. This demands an efficient search strategy that combines various knowledge sources.

In this thesis, we present solutions to the above problems, which will provide a new framework of speech recognition suitable for spoken language systems and dialogue systems.

1.1 What is Necessary

In order to realize an intelligent and user-friendly spoken language system or dialogue system, the following factors are significant.

Speaker-independent natural speech

Speaker-independence is essential because spoken language systems must be available to any users. Some applications such as guidance or booking should be open to the general public. Even if training with a new speaker is permissible, the baseline system has to be powerful enough to minimize the necessary training samples.

Conversational speech spoken to a dialogue system is different from inputs to a dictation machine or voice typewriter. A user tries to communicate his or her intention by putting stress on keywords or changing speeds, which causes unclear words on the other hand.

To meet these needs, the acoustic model that is the base of the whole system must have high discriminating power. It also has to be robust against both the inter-speaker variation and the poor articulations.

Continuous utterance with sufficient perplexity

Accepting continuous utterances is also vital factor in spoken language systems. It is unnatural and even painful for a user to cut conversational speech into words. It is a heavy burden especially on speakers of Japanese, as the concept and the boundary of words are not explicit. When a system does not know the word boundary a priori by allowing continuous utterances, it has to assume the word boundary at every frame. Thus, the number of sentence hypotheses that the system deals with increases drastically.

Moreover, in real applications, a variety of utterances must be accepted and the perplexity of tasks gets large. This is another factor that causes the number of hypotheses to explode.

These factors require a search algorithm that can find the optimal hypothesis with high accuracy and less computation.

Robust understanding of ill-formed utterances

Unlike read speech to a dictation system, there are inevitably various ill-formedness in spontaneous speech that a spoken dialogue system has to deal with. Hesitations and filled pauses such as 'uh' are often inserted, and even violation of the orthodox grammar such as an incorrect use of functional words occurs. The conventional natural language analyzers hardly cope with them.

New words unknown to the system are also often used. They upset the conventional recognizers that presume a prepared vocabulary. Even with a large vocabulary system, names of cities and persons are not predicted completely.

Robustness against these sorts of ill-formed utterances is significant to make the system practical in real situations. In this case, the system only has to understand utterances, and complete transcriptions of them are not necessary.

1.2 Approach

To address the above issues, the following approaches are adopted in the thesis.

1.2.1 Discriminative and Robust Acoustic Modeling

At first, the acoustic model must be improved to be discriminative and robust since it is the basis of the overall system.

Discriminative ability means to what extent the recognizer can classify the given samples that are provided for training. Robustness means to what extent the recognizer can cope with open samples that are different from training samples.

Discriminative ability and robustness can be in a trade-off relation generally. For example, in the connectionist approach, one big problem is overlearning, where a neural network adapts too much to training samples and does not work on open samples such as those of different speakers[14]. Model complexity is another related issue. As the number of model parameters increases, potential capacity of complex classification increases. But it gets hard to estimate the parameters accurately and hard to construct a general and robust model[15].

As acoustic modeling, HMM is most promising. It models intra-speaker and inter-speaker variation with probabilistic output distributions and realizes dynamic time-warping with probabilistic state transitions. It can be regarded as a generalization of DP matching. Although HMM provides a comprehensive framework, its implementation gives rise to some significant problems. As for the probability evaluation problem and the state sequence decoding problem, Forward algorithm and Viterbi algorithm give optimal solutions respectively. However, the model construction problem, including a front-end quantizer in discrete model, and the training problem remain. The orthodox HMM formulation lacks the concept of discriminating competing classes, since it only constructs and estimates a phone model independently, using the samples that belong to the phone category.

In this thesis, we study how to improve HMM for both discriminative ability and robustness. The approach we adopt is discriminant analysis which reduces the parameters and improves the separability between the classes. Reducing the number of parameters makes their estimation accurate with a reasonable number of samples, and also improves robustness. Furthermore, we present a more sophisticated model that is based on pair-wise discriminant analyses, since distinctive features vary for different phones and different discriminant functions should be made for different competing classes.

First, we apply this approach to discrete distribution HMM by making the set of observation symbols best separated, and then to continuous distribution HMM by best separating the states of the different phone models.

1.2.2 Admissible and Efficient Search Algorithm

The search algorithm generates and evaluates plausible sentence hypotheses to get the optimal result among a vast number of possible hypotheses. There are two points which must be taken into account: admissibility and efficiency.

Admissibility means that a search algorithm is guaranteed to find the optimal hypothesis whose score is best under given constraints. Efficiency is measured by the number of hypotheses that an algorithm generates and extends. Both requirements may be in trade-off relation. Pursuing efficiency often forces us to give up admissibility.

We first put priority on admissibility, because we cannot seek and evaluate efficient algorithms unless we know the upper bound of the accuracy. But this does not mean that we study unrealistic search algorithms that cannot be implemented with currently available computing resources.

Incorporating good heuristics into a search is effective to satisfy both admissibility and efficiency. If the heuristics is A*-admissible, the search finds optimal hypothesis. If the heuristics is powerful enough, it guides the search to the solution efficiently. Good heuristics is the key, considering its dedicated computation besides the search itself.

In this thesis, we focus on realization of this kind of heuristic search. Once we realize a heuristic search that is admissible, then it is possible to extend it for further efficiency by adding such techniques as beam pruning.

First, we present an A*-admissible LR parsing algorithm for continuous speech recognition. Then, for more robust recognition of spontaneous speech, we study word spotting and phrase spotting algorithms based on the framework of heuristic search.

1.2.3 Integration of Statistical and Symbolic Knowledge

Language modeling is another component of speech recognition. Its function is to provide the speech recognizer with predictive or verifying information to limit the hypotheses, namely to constrain the search space.

When the goal of recognition is dictation, the knowledge source which is directly related to recovering the surface word sequences is preferable. In spoken dialogue systems, however, a different viewpoint on language modeling is necessary. As the goal of spoken dialogue system is to understand user utterances, it needs a natural language processing system to get the most plausible interpretation, using linguistic and pragmatic knowledge according to the context and the situation. It is desirable for a front-end speech recognizer to be linked with the understanding process.

There are two sorts of knowledge representations: statistical and symbolic.

A statistical language model N-gram has been widely used since it implicitly embeds syntactic, semantic and pragmatic information. It is hard to describe general knowledge explicitly. The simplicity of N-gram is also advantageous in the computational aspect. But there exists a certain limitation of the model due to the absence of the concept of understanding.

As a symbolic knowledge representation, grammar-based approach has also been studied. It uses a simple grammar dependent on the task domain. It can incorporate longer distance constraint than bigram or trigram that concerns only one or two preceding words, although it is less robust. Above all, a grammar can extract the structure of utterances which is directly mapped to a semantic representation.

Our approach is integrating these two. In order to decode an input to a word sequence, a statistical language model is more effective and robust. To lead higher-level understanding of the speech, a grammar of structural and symbolic knowledge is incorporated.

The integration is realized in the framework of heuristic search described in the previous subsection. It is formulated as follows. A partial sentence or phrase hypothesis n is evaluated by the sum of the score given by the both language models.

$$f(n) = g(n) + h(n) \tag{1.1}$$

$g(n)$: score of the parts parsed by a grammar

$h(n)$: score of the rests constrained by a statistical model

Here, the heuristics given by a statistical language model $h(n)$ provides perspective of the hypothesis, which guides the symbolic parser.

The mechanism is realized as a progressive search, which first applies a statistical language model, and then performs symbolic parsing using the result of the first pass as heuristics.

1.2.4 Cooperation of Speech Recognizer and Natural Language Analyzer

In the conventional spoken language understanding systems, a speech processor and a natural language processor are combined in cascade. They are fundamentally separated and the speech recognizer passes its output such as sentence hypotheses to the language analyzer. This simple mechanism keeps the modularity of the two processors with little interface, which enables easy development and maintenance. This paradigm is called the N-best paradigm, which first applies simple constraints and then examines the output sentence candidates with complex language processing.

However, when the two modules are separated, higher-level linguistic knowledge sources such as semantic and dialogue-level ones cannot be applied at the speech recognizer. So the errors at the recognition stage cannot always be corrected. Concretely, the N-best sentence candidates obtained with loose constraints tend to be quite similar, having the same syntactic and semantic structure with one or two words replacements[16]. Thus, the linguistic analysis following it cannot recover the structural recognition errors.

On the other hand, a tight integration of the two processors is not easy, since the characteristics and the purpose of the knowledge sources are fundamentally different. The speech recognizer uses knowledge of lexicon and syntax to constrain the surface word sequence, while the natural language understanding process needs conceptual-level knowledge sources such as semantics and pragmatics. These different knowledge sources are not easily transformable.

We propose, therefore, not direct integration but interaction of the knowledge sources, that is cooperation of the speech recognizer and the natural language analyzer. While keeping the modularity, the speech recognizer and the language processor interact with each other to exchange constraints.

The model is illustrated in Figure 1.1. The speech recognizer performs a tree search

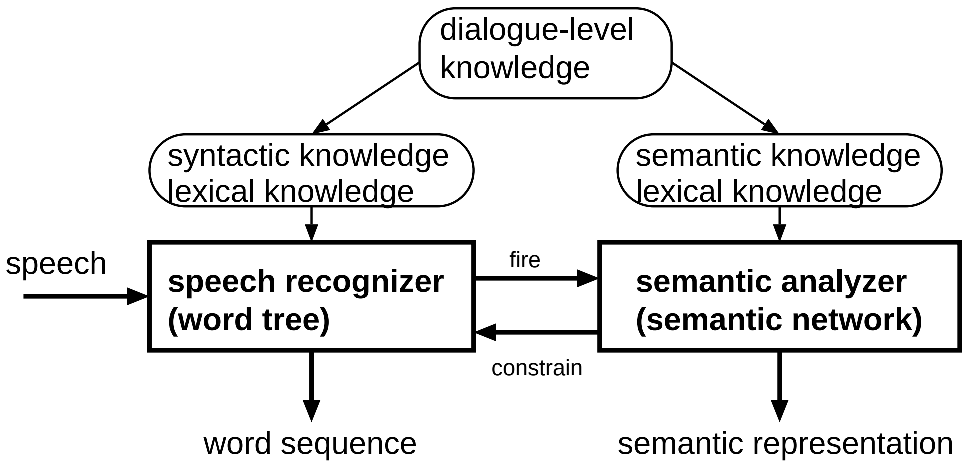
dialogue-level knowledge
syntactic knowledge lexical knowledge
semantic knowledge lexical knowledge
speech
speech recognizer (word tree)
fire
constrain
semantic analyzer (semantic network)
word sequence
semantic representation

Figure 1.1: Cooperative speech understanding model

driven by a grammar, which generates a corresponding HMM trellis. The search finds the optimal word sequence mainly with lexical and syntactic knowledge sources. The natural language processor searches for the most plausible semantic representation, using lexical and semantic knowledge sources. All the knowledge sources the both processors use are affected by dialogue-level knowledge. For example, the syntactic type of the current utterance reflects its intention in a stream of the dialogue, and the vocabulary is constrained by the current topics and focus. Moreover, the two processors interact with each other to constrain the counterpart search. Concretely, the intermediate results of the word tree search fires the semantic-level search. In turn, the semantic analyzer interactively constrains the word tree. As a result, all the knowledge sources are exchanged.

There are two ways to realize this cooperative understanding model of the speech recognizer and the semantic analyzer. Although the both proceed interactively, one of them takes initiative to construct sentence hypotheses. When we adopt a stack decoder for overall control, the master processor directly accesses the stack, predicting the possible hypotheses. The other or slave processor works as a verifier. Normally, the speech recognizer is master and constructs the sentence hypotheses, which are verified by the semantic analyzer incrementally. But it is also possible for the semantic analyzer to be master. In that case, sentence hypotheses are predicted by the semantics and evaluated by the speech recognizer.

1.2.5 Parsing versus Spotting

As the linguistic analysis for spoken language understanding, there exist two approaches: parsing and spotting. Parsing approach models the whole part of the utterance and describes as many phenomena as possible. It tries to analyze the whole given input. Spotting approach focuses on keywords or key-phrases that significantly contribute to sentence understanding. It extracts such parts and skips the rest of the input.

In the conventional spoken language understanding systems, parsing approach is popular. The constraint given by a language model is applied to the whole utterance and the result is obtained after the global optimization. But this approach cannot cope with what is not modeled or described, though it tries to parse the whole input. Moreover, the constraint of a lexicon and a grammar, which is very powerful in recognition, puts a burden on users by restricting the vocabulary and the patterns of sentences. And it would be very hard and possibly endless to describe a grammar that covers various flexible utterances without increasing perplexity.

As for robustness against ill-formed utterances, spotting approach is attractive. It extracts only recognizable parts and skips the rests that cannot be not modeled. Actually, in our daily life, especially in a limited situation, we can mostly make senses of utterances by putting attention to keywords and discarding the rest detailed parts. However, the spotting basically performs length-free matching with weak constraint, which involves substantially large perplexity. It might be critical to make recognition without linguistic knowledge.

In this thesis, we study the both approaches, seeking to improve them respectively. Finally, they are compared with regard to accuracy and robustness.

1.3 Outline of the thesis

In Chapter 2, we make an overview of speech recognition. The design of the spoken dialogue system of our group and the role of speech recognition in the system are presented. Then, the recognition processes including signal processing, acoustic model, and language model are briefly examined. Their basic concept, the conventional algorithms and their defects are explained. The task and the database used in the thesis are also described.

In Chapter 3, we discuss the discriminative statistical model for the acoustic model that is the basis of the whole recognition system. Both discriminative ability and robustness are explored.

In Chapter 4, we present admissible and efficient heuristic search algorithms for continuous speech parsing. The integration of the multiple knowledge sources and the cooperation of the speech recognizer and the natural language analyzer are also discussed.

In Chapter 5, we present word and phrase spotting algorithms with heuristic language models. Here also, statistical knowledge and symbolic knowledge are integrated. Then, the spotting approach is compared with the parsing approach.

Chapter 6 concludes the thesis.

Chapter 2

Overview of Speech Recognition

2.1 Outline of Dialogue System

2.1.1 System Design

As an intelligent and user-friendly human-machine interface, we (Prof. Doshita's Lab.) are designing and implementing a spoken dialogue system. Its platform is a personal desktop workstation. It features a new interactive mode to the computer, using a microphone and a speaker. The system understands user utterances and makes responses to them appropriately. By continuing a dialogue, it responds to the user's demand.

We have set a task domain of personal schedule management. Its concept is a 'computer secretary', which checks the user's schedule and makes liaison with others via computer networks. Present design is just a schedule database management, which includes registration, modification and query of schedule data. It is regarded as a kind of intelligent database access with spoken language. So the goal of the system is to interpret the user's demand through a dialogue into a command of the database interface.

This system is to accept continuous utterances by any speakers. However, we now impose several constraints. At one dialogue session, just one demand which corresponds to a database access command can be realized, though it may branch to another subdemand on its way. One utterance must be composed of single sentence, though it may be broken to some extent.

The spoken dialogue system is an integration of speech processing and natural language processing as well as knowledge processing. Figure 2.1 shows its flowchart. The uppermost part is a user, who makes utterances to and gets responses from the system.

The left part surrounded by a broken line corresponds to speech understanding process.

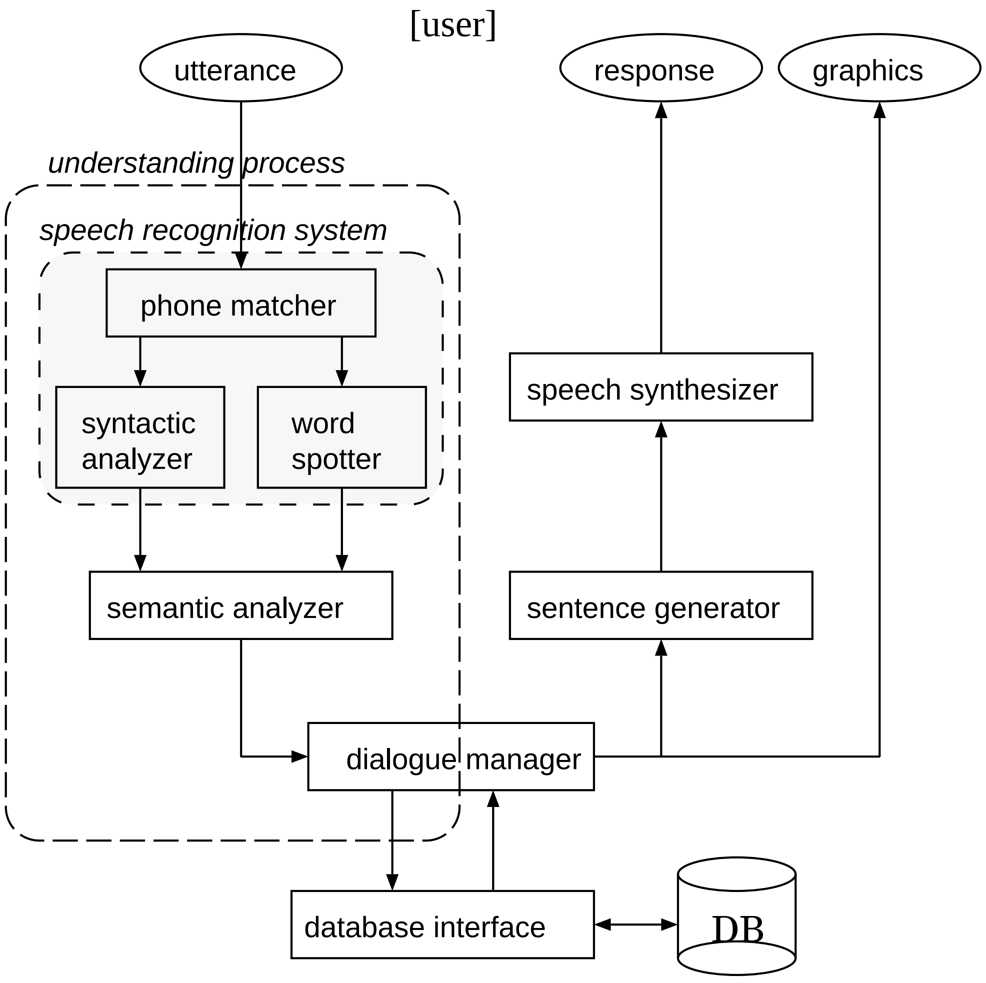

Figure 2.1: Flowchart of dialogue system

It inputs user utterances and extracts their meanings, considering the context of the dialogue. If it succeeds to generate a database access command, it calls the database interface program. The dotted area is a speech recognition system, on which the thesis studies mainly. Other area in the understanding process is called a natural language understanding system.

The right part is responding process. A dialogue manager controls generation of responding utterances, which are output to the user through speech synthesizer. Responses to the user are also done through the graphic user interface if appropriate.

2.1.2 Flow of Understanding Process

In order to understand user utterances, we have adopted two different approaches that are introduced in Chapter 1. The two paths between the phone matcher and the semantic analyzer in Figure 2.1 corresponds to these approaches respectively.

Parsing Approach

This approach performs parsing on the whole inputs with some grammar and obtains the N-best sentence candidates, which, by turns, are passed to semantic analyzer to construct a semantic representation. It requires a user to utter strictly grammatical sentences, which is strong constraint but expected to improve recognition accuracy. The function of each process is explained in order.

1. Phone matcher

   The phone matcher is a front-end processor that inputs user utterances and performs phoneme recognition. However, it does not work alone, as phoneme recognition without any linguistic constraints is almost impossible. It is driven by the syntactic analyzer. Namely, the parser predicts some word, which is expanded to a phone sequence, and asks the phone matcher to evaluate it. So the output of the matcher is a score of the predicted phone sequences.

2. Syntactic analyzer

   The syntactic analyzer is the core in parsing approach. It constructs sentence hypotheses for an input utterance. It predicts possible partial sentences, calls the phone matcher, and extends the plausible ones based on the matching results, until a complete sentence is obtained. By continuing this process, finally, the N-best sentences are output to the semantic analyzer.

3. Semantic analyzer

   The semantic analyzer constructs semantic representations for given user utterances. In parsing approach, it analyzes syntactically complete sentences in turn from the best candidate, until a correct semantic representation is obtained. Therefore, conventional parsers for written language can be used with minor modifications.

4. Dialogue manager

   The dialogue manager plays important roles in understanding utterances. It fills and even modifies some slots of a semantic representation according to the context of the dialogue. It also provides predictive information to the semantic analyzer and the syntactic analyzer. Use of this information in the syntactic level is discussed later.

Spotting Approach

This approach performs word or phrase spotting and obtains a set of word/phrase candidates with their scores. If they are time-aligned at this stage, the set is called a word/phrase lattice. They are passed to the semantic analyzer. In this case, the semantic analyzer must parse a set of words/phrases, not complete sentences. This approach allows more flexible utterances that do not have to follow grammars. It proceeds as follows.

1. Phone matcher

   Same as in parsing approach, except that it is driven by the word/phrase spotter.

2. Word/phrase spotter

   The word/phrase spotter is a pre-processor of the approach. It recognizes words or phrases with their scores and time-alignment information. It calls the phone matcher and extends the plausible word/phrase candidates. A set of candidates whose scores exceed some threshold is output.

3. Semantic analyzer

The role of the semantic analyzer is fundamentally same as that in parsing approach. In spotting approach, however, it constructs sentence hypotheses from a set of words/phrases. It predicts possible combinations of words/phrases and evaluates their scores and the coverage of the input speech. Then plausible ones are extended, until a semantic representation covering reasonable parts of the speech is obtained.

4. Dialogue manager

   Same as in parsing approach.

2.2 Components of Speech Recognition

Next we clarify how speech recognition, the thesis's theme shown as dotted area in Figure 2.1, is carried out. Figure 2.2 shows the more detailed flow of speech recognition. The upper part is the phone matcher and the lower is the syntactic analyzer and the word/phrase spotter.

A user utterance is input to the signal processing part, which extracts useful features for recognition. The obtained sequence of pattern vectors is passed to the pattern processing part, which is based on acoustic modeling. The phone matcher is tightly integrated with language modeling, which describes linguistic knowledge and provides predictive information. A search algorithm is the key to combine acoustic modeling and language modeling. It constructs hypotheses based on the predictive information from the language model and matches them with the acoustic models for scoring. Then it extends plausible hypotheses by continuing this process. The required output is sentence candidates or a set of words or phrases. Although algorithms are different for these purposes, the fundamental concept of searching is same.

The basic approaches and methods of each component that are adopted in the study are explained.

2.2.1 Signal Processing

Speech analysis with digital signal processing is performed in order to extract features that are speaker-independent and distinctive of phones. Typically, a power spectrum or its equivalent such as a cepstrum is obtained as a pattern vector, which is input to the following recognition process.

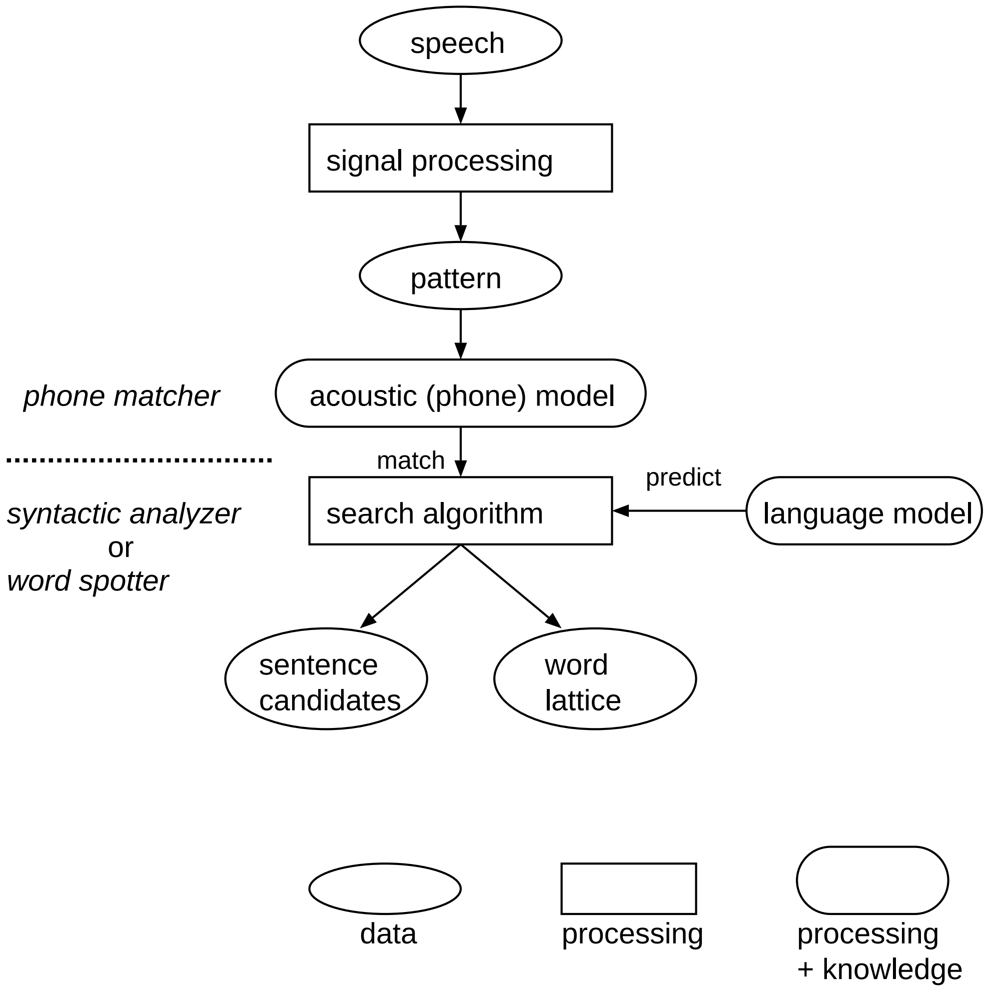

Figure 2.2: Flowchart of speech recognition system

As the quality of the pattern affects the whole recognition, a lot of studies on speech analysis have been undertaken. The most successful methods so far are based on linear predictive coding (LPC).

In our experiments, a speech sample is sampled at 16 kHz and quantized to linear 16 bit. The speech waveform is then segmented into frames. The frame length is 25 ms. The frame shift or the interval of analysis is 10 ms. Hamming window is applied to each frame. For each windowed frame, LPC analysis of 26th order is performed. We compute from the coefficients a smoothed power spectrum of 28 channels that correspond to the critical band-widths. Beside this, we compute a power parameter with the residue of LPC analysis.

In recognition, it is pointed out that the dynamic feature is significant such as delta-cepstrum[17]. Here, we adjoin 8 frames of above variables into a pattern vector[18]. Since its dimension gets quite large, discriminant analysis described later is performed to select effective variables and eliminate redundant ones.

2.2.2 Acoustic Modeling

An input speech pattern, through the analysis, is matched with acoustic models. The orthodox pattern recognition methodology consists of formation of templates to be matched and definition of a distance to evaluate the input. The most trivial case is combination of the means of the training samples and Euclid distance.

In speech recognition, there arises the problem of dynamic time-warping of the matching, as the length of the pattern is not fixed and it is far from linear warping. Therefore, dynamic programming (DP) is widely used to find the best matching sequences. Another problem is treatment of speaker variation. Conventional way is to have multiple templates. But this approach needs not a few number of parameters.

The more sophisticated approach is stochastic one. It models speaker variation with a statistical distribution and realizes dynamic time-warping with a probabilistic process. This model is called hidden Markov model (HMM), as its underlying states that have statistical distributions transit in a Markov process. It is also regarded as a probabilistic non-deterministic automaton, which generates or accepts a speech parameter sequence with some probability. Though it models the speech generation process, in recognition, it works as an acceptor of input sequences. It is illustrated in Figure 2.3. As generation of speech is not reversible process, it is usually modeled with left-to-right model. In

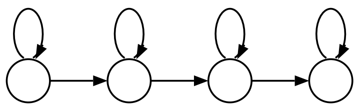

Figure 2.3: Structure of left-to-right HMM

some cases, transitions skipping some state are used, but they need extra computation with little effect. There is a type of automaton, where an output distribution is attached to the state transition instead of the state itself. It needs more parameters and it gets equivalent to Moore type by tying the distribution of the same states. Notice also that HMM is equivalent to a DP template by allotting the same states as the template's time-slots and making the same state transitions with even probabilities.

This sort of model is constructed and trained for each recognition unit such as a phoneme. For a given input $\mathbf{X}$, we recognize it into the model m (or concatenation of models in word/sentence recognition), such that probability $p(\mathbf{X}|m)$ that $\mathbf{X}$ is generated by m is highest.

Forward Algorithm for Recognition

To compute the probability $p(\mathbf{X}|m)$, the following forward algorithm is performed.

For a model m of n states that starts at s_1 and ends at s_n, and an input pattern sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, we perform an iterative computation.

$$\begin{aligned}
\alpha_1(1) &= p(\mathbf{x}_1|s_1),\ \alpha_1(i) = 0 \qquad (2 \le i \le n) && (2.1)\\
\alpha_t(i) &= \sum_j \{\alpha_{t-1}(j) \cdot p(s_i|s_j)\} \cdot p(\mathbf{x}_t|s_i) \qquad (1 \le i \le n, 2 \le t \le T) && (2.2)\\
p(\mathbf{X}|m) &= \alpha_T(n) && (2.3)
\end{aligned}$$

Here $p(s_i|s_j)$ is a state transition probability from s_j to s_i. It has to satisfy the following relation.

$$\sum_i p(s_i|s_j) = 1 \tag{2.4}$$

And $p(\mathbf{x}_t|s_i)$ is an output probability that a pattern $\mathbf{x}_t$ comes from state s_i. In the continuous distribution model, we usually assume Gaussian distribution, which is characterized with a mean μ and a covariance $\boldsymbol{\Sigma}$. More elaborate model adopts a weighted combination of multiple distributions that correspond to multiple templates.

$$p(\mathbf{x}_t|s_i) = C \cdot \exp\{-(\mathbf{x}_t - \mu_i)^t \cdot \boldsymbol{\Sigma}_i^{-1} \cdot (\mathbf{x}_t - \mu_i)/2\} \tag{2.5}$$

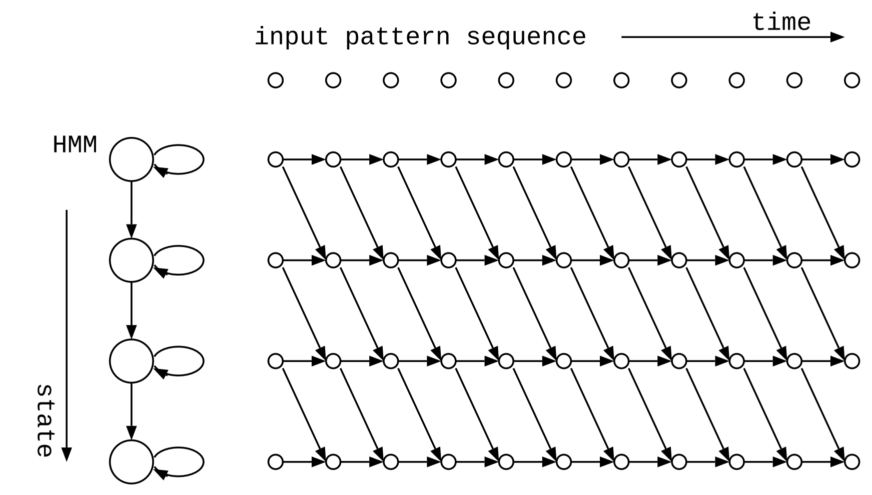

Figure 2.4: HMM trellis

$$C = (2\pi)^{-d/2} \cdot |\Sigma_i|^{-1/2} \tag{2.6}$$

In the discrete distribution model, we firstly quantize a pattern $\mathbf{x}_t$ into a code or symbol o_t, and the output probability is defined for each code. In this case, probability computation is realized as a table look-up.

$$p(\mathbf{x}_t|s_i) = p(o_t|s_i) \tag{2.7}$$

If we assume a simple left-to-right model without skipping transitions, equation (2.2) becomes:

$$\alpha_t(i) = \{\alpha_{t-1}(i) \cdot p(s_i|s_i) + \alpha_{t-1}(i-1) \cdot p(s_i|s_{i-1})\} \times p(\mathbf{x}_t|s_i) \tag{2.8}$$

With this assumption, the algorithm is realized as a simple trellis computation shown in Figure 2.4.

Viterbi Algorithm for Recognition and Decoding

As an approximating method, Viterbi algorithm replaces the iteration with following.

$$\alpha_t(i) = \max\{\alpha_{t-1}(i) \cdot p(s_i|s_i),\ \alpha_{t-1}(i-1) \cdot p(s_i|s_{i-1})\} \times p(\mathbf{x}_t|s_i) \tag{2.9}$$

By making $\alpha_t(i)$ logarithm scaled, this computation becomes the efficient dynamic programming without multiplication.

$$\begin{aligned}\log \alpha_t(i) = \quad & \max \quad \{\log \alpha_{t-1}(i) + \log p(s_i|s_i), \\ & \qquad\quad \log \alpha_{t-1}(i-1) + \log p(s_i|s_{i-1})\} \\ & + \log p(\mathbf{x}_t|s_i)\end{aligned} \tag{2.10}$$

To estimate the most likely state sequence $Z = (z_1, \ldots, z_n)$, namely to decode the input with the HMM states, we simply have to remember the pointer $\phi_t(i)$ that gives maximum at every frame and trace back from the endpoint.

$$\phi_t(i) = \begin{cases} i & \alpha_{t-1}(i) \cdot p(s_i|s_i) > \alpha_{t-1}(i-1) \cdot p(s_i|s_{i-1}) \\ i-1 & \text{else} \end{cases} \tag{2.11}$$

$$z_T = n \tag{2.12}$$

$$z_t = \phi_t(z_{t+1}) \qquad (T-1 \geq t \geq 1) \tag{2.13}$$

Forward-Backward Algorithm for Training

In order to train the model, namely to estimate the HMM parameters, we introduce a backward probability.

$$\beta_T(n) = 1,\ \beta_T(i) = 0 \qquad (1 \leq i \leq n-1) \tag{2.14}$$

$$\beta_t(i) = \sum_j \{p(s_j|s_i) \cdot p(\mathbf{x}_{t+1}|s_j) \cdot \beta_{t+1}(j)\} \qquad (1 \leq i \leq n, T-1 \geq t \geq 1) \tag{2.15}$$

The state transition probability $p(s_j|s_i)$ is updated as follows.

$$\hat{p}(s_j|s_i) = \frac{\sum_t \alpha_{t-1}(i) \cdot p(s_j|s_i) \cdot p(\mathbf{x}_t|s_j) \cdot \beta_t(j)}{\sum_t \alpha_t(i) \cdot \beta_t(i)} \tag{2.16}$$

The output probability for each code entry in discrete model is also updated.

$$\hat{p}(o_t = k|s_i) = \frac{\sum_{t:o_t=k} \alpha_t(i) \cdot \beta_t(i)}{\sum_t \alpha_t(i) \cdot \beta_t(i)} \tag{2.17}$$

In the continuous distribution model, both means and covariances are updated.

$$\hat{\mu}_i = \frac{\sum_t \alpha_t(i) \cdot \beta_t(i) \cdot \mathbf{x}_t}{\sum_t \alpha_t(i) \cdot \beta_t(i)} \tag{2.18}$$

$$\hat{\Sigma}_i = \frac{\sum_t \alpha_t(i) \cdot \beta_t(i) \cdot (\mathbf{x}_t - \mu_i) \cdot (\mathbf{x}_t - \mu_i)^t}{\sum_t \alpha_t(i) \cdot \beta_t(i)} \tag{2.19}$$

We continue these reestimations until they converge. This kind of iteration is known as EM (Expectation-Maximization) algorithm and guaranteed to converge into a local optimal solution.

Several Approaches to Improve Modeling and Training

There are several approaches to improve the orthodox HMM that is formulated above.

(1) Refining the phonetic unit to context-dependent

The approach divides the phonetic unit into several clusters, considering its neighboring phonetic contexts[19],[20]. As the context-dependent clusters tend to have respective small distributions, overall separability between different phones is expected to be improved. This approach creates so many models and drastically increases the model parameters, which faces the model complexity problem. It tries to solve by using enormous number of training samples. The increase of the number of models also makes the linguistic parser complex, generating so many grammar nodes.

(2) Elaborating the model structure

The approach makes the model to describe more in detail. One way is to increase the number of states, adding complex state transitions[21]. Another way is to refine the output probability distribution, that is to increase the codebook entries in the discrete distribution model, and to increase the number of mixtures in the continuous distribution model. This approach also increases the model parameters on the other hand.

(3) Discriminant analysis of feature vectors

The approach is to modify the pattern space of the input feature vectors so that the transformed space is better separated. Techniques such as discriminant analysis and subspace projection are used for transforming the pattern vectors to improve the separability between the symbols of the discrete model or the HMM states of the continuous model. This approach does not increase the model parameters, rather reduces them.

(4) Discriminative training

The approach focuses on learning and discrimination scheme. While the conventional Forward-Backward algorithm estimates the parameters of a model using in-class samples only, this approach use the samples of all classes totally to improve separability of them. Alternative criteria have been proposed such as maximum mutual information[22], general probabilistic descent[23] and corrective training[24] These sorts of supervised training are very effective in improving discriminating ability. But they may fall in the overlearning problem, and lack of robustness against the open samples

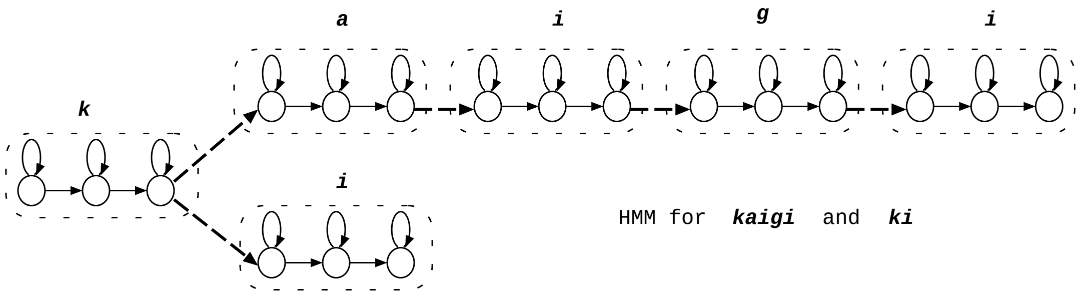

Figure 2.5: Concatenating HMMs to represent a word/sentence

Although all of the above will be useful and they can be combined each other, we focus on the discriminant analysis of feature vectors as it will realize the most substantial improvement without losing robustness, with a reasonable number of training samples.

2.2.3 Language Modeling

With an acoustic model, we obtain the best word/sentence candidate W based on probability $p(\mathbf{X}|W)$. The probability is computed by concatenating phone HMMs into a big model to represent the hypothesis as shown in Figure 2.5.

A language model plays two roles in speech recognition. The first one is to predict valid phone sequences and eliminate those which are not accepted by linguistic and pragmatic constraints. It reduces the number of candidates W to be matched with the phone model. Without this constraint, the number of hypotheses increases in the exponential order. The second role is to give preference to the candidates. It causes more plausible candidate to be chosen even if its matching score is lower. This mechanism is formulated by Bayesian rule to get a posteriori probability.

$$p(W|\mathbf{X}) = \frac{p(W) \cdot p(\mathbf{X}|W)}{p(\mathbf{X})} \tag{2.20}$$

Here $p(W)$ is a priori probability given by a language model. When we remove $p(\mathbf{X})$, which does not affect the recognizing decision, and take logarithm as in Viterbi computation, the equation becomes a summation of the score by the acoustic model and that by the language model. The probability given by the language model is based on either statistics or specific situation.

$$\log p(W|\mathbf{X}) = \log p(W) + \log p(\mathbf{X}|W) \tag{2.21}$$

There are two approaches to language modeling. One is thoroughly statistical and the other is knowledge-based.

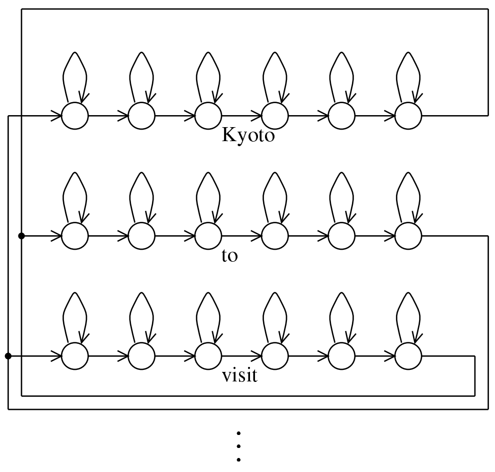

Figure 2.6: Word bigram model

Statistical Language Modeling

The statistical approach models language with Markov models of words or morphological elements. With simple bigram, a priori probability for word sequence $W = (w_1, \ldots, w_l)$ is:

$$p(W) = p(w_1|.)p(w_2|w_1)\ldots p(.|w_l) = \prod_i p(w_{i+1}|w_i) \tag{2.22}$$

Word trigram can be also used as a higher-level constraint.

In Japanese, the concept and the boundary of words is not explicit and words are not separated in normal texts. Therefore, it is not easy work to estimate the word bigram. Automatic estimation with large corpora needs a morphological analyzer, which is not perfect so far. Therefore, syllable bigram can be alternative. Japanese syllable that consists of a vowel preceded by a consonant corresponds to an elementary character called kana.

The statistical language modeling reflects all linguistic and pragmatic constraints specific to the task domain. When the purpose of the language model is decoding the original word sequence from the inputs such as dictation or translation, it is quite effective. The word bigram model, as shown in Figure 2.6, is also advantageous from the computational aspect since it is fundamentally of the same order of the vocabulary size.

However, it is almost impossible to extract higher-level information such as the meaning of utterances with this model, thus it does not contribute to the language understanding process. Therefore, we basically use it for heuristics for parsing and spotting that is combined with the knowledge-based language model.

Knowledge-based Language Modeling

In order to form a knowledge-based language model, then, we examine available knowledge sources. In the conventional linguistics, lexical, syntactic, semantic and pragmatic knowledge sources are used. In a dialogue system, the dialogue-level constraint can be also used. Furthermore, we suppose task domain-specific constraint which is reflected in all levels mentioned above.

Lexical knowledge is most fundamental and computationally cheapest. It is represented as a dictionary that transcripts phonetic sequences of words.

Syntactic knowledge is significant in sentence recognition. Several classes and types of grammars are conventionally used. The simplest one is a word-pair grammar that describes which words can follow some word. It is a special case of the word bigram. An automaton grammar or a network grammar is widely used as it is straightly expanded to HMM state sequences. But it is highly troublesome to construct this kind of representation, especially in a large task. As for a context-free grammar, which is based on syntactic rewriting rule sets, several parsers have been studied. Early's algorithm[25] is well known as a top-down parser, and CYK chart-based algorithm[26] is a famous bottom-up parser. Both of them are simple and powerful, but generates all the possible hypotheses, many of which are in vain. Therefore, it is difficult to apply them to a large grammar. The most efficient parser so far is LR parser such as Tomita's algorithm[27]. It transforms the rule sets into state tables. As a word is input, it transits the states using a stack to save and restore the status. Predicted words at the state are specified in the table.

Since a context-free class grammar is desirable and convenient to describe normal conversational language, we use LR parser. But we do not commit to concrete implementation of the parser. It is just required, for a given input, to check if it is accepted as a complete sentence and to predict next word candidates. Furthermore, we use lower-level knowledge sources as heuristics to guide the context-free parser.

With rules only, possible but less likely sentence hypotheses are generated as equally as frequent sentence patterns. In order to reflect statistics, probability is introduced to rank the rules so that more likely sentence hypotheses are adopted. This probabilistic grammar gives a score of the language model in equation (2.21). The probability is attached to each rewriting rule, so that the sum of those that have a same left non-terminal symbol becomes 1. Its example is shown in Figure 2.7.

	(prob.)
PERIOD_PP → DAY_N "kan"	0.2
PERIOD_PP → TIME_N "kan"	0.7
PERIOD_PP → TIME_N "kan" "han"	0.1

Figure 2.7: An example of probabilistic grammar rules

Semantics, pragmatics and the dialogue-level knowledge source are mainly utilized in natural language analyzer. However, semantics and pragmatics are actually embedded more or less to the grammar used in speech recognition since it is useful for disambiguation and reducing perplexity. Use of the dialogue-level knowledge source in the speech parser is also discussed later. But the basic concept of our model is that these knowledge sources are indirectly accessed by interaction with the semantic analyzer.

2.2.4 Search Algorithm

Given an acoustic model and a language model, the number of hypotheses becomes enormous in continuous speech recognition. For example, when the average number of predicted words in each extension is 10 and the possible length of a sentence is 7, then total number of generated sentences can be more than 10 million.

When the number of grammar states is small such as word-pair, Viterbi search is possible. It performs Viterbi algorithm on the model that concatenates phone HMMs to represent the grammar, thus finds the optimal hypothesis by counting only path that gives a maximum score at every frame. It realizes full search.

As a grammar becomes complex containing so many states, it costs too much to compute the scores for all states. In context-free grammar, which has recursive nodes, it is theoretically impossible to compute all. In this case, phone models are dynamically concatenated as a hypothesis grows, and the trellis is generated accordingly.

For a large grammar, therefore, some pruning or ordering technique is essential. In search algorithm, there are several issues to be considered: the order of hypothesis generation, the evaluation function, and the direction of search.

Best-first vs. Beam

With regard to the order of hypothesis generation, best-first search and beam search are mainly used in speech recognition.

Best-first search extends the hypothesis whose score is highest. However, the search possibly falls into a breadth-first state, which generates so many hypotheses but does not proceeds. To avoid this, stack decoding[28] is used to limit the number of hypotheses. Stack is an efficient mechanism to push generated hypotheses and to pop the currently best hypothesis.

Beam search has both features of breadth-first and best-first search. It proceeds breadth-first, namely extends all the current hypotheses, but eliminates those of lower scores in order to limit the number of hypotheses within a certain beam width. In other words, it extends N-best (for large N) hypotheses at every step.

Beam search can be implemented as both time synchronous and asynchronous, while best-first search proceeds asynchronously. However, stack decoding may fall into the much same state as beam search, if the evaluation function highly depends on the time length of the hypothesis. This implementation problem makes beam search more popular, though stack decoding has attractive features. Therefore, the evaluation function is examined to realize an effective best-first search.

Blind vs. Heuristic

As for the evaluation function or scoring, the simplest and currently most popular method is based on Viterbi score for the generated hypothesis. It assumes time-synchronous search, otherwise the score of longer hypothesis gets lower. It implies that the search is blind, where all the hypotheses are extended without estimating how probably they will bring out the optimal path in the end.

On the other hand, there is an approach to incorporate heuristic score into the evaluation function. Heuristic search estimates the score of the remaining unsearched part and adds it to the current score. Its evaluation function for a hypothesis n is formulated as follows.

$$\hat{f}(n) = g(n) + \hat{h}(n) \tag{2.23}$$

Here $g(n)$ is a matching score from the beginning of the input speech to the searched point, and $\hat{h}(n)$, called heuristic function, is an estimate of the best scoring extension to the end of the input. The search is A*-admissible, namely guaranteed to find the optimal path, if the estimate is an upper bound of the actual scoring extension of the hypothesis to the end, that is $\hat{h}(n) \geq h(n)$.

When $\hat{h}(n) = 0$, then the search is blind. However, it is not easy to evaluate the heuristic score accurately. If it is too optimistic, the search falls into blind. Therefore, several approaches underestimate the score to proceed search effectively. But this invalidates the admissibility and may lose the optimal path. Our study concentrates on this issue in Chapter 4.1.

There are several other techniques to improve the search. When the vocabulary size gets large, fast match that tests small fragments is effective to choose the word to extend the hypothesis.

Island-Driven vs. Left-to-Right

As for the direction that a search proceeds, one-directional search is popular. Especially, left-to-right search is suitable for real-time processing. Island-driven search has also been studied. It starts with reliable words and extends hypotheses to the both directions, skipping noisy parts. It is not so popular because its implementation gets complex and the number of generated hypotheses gets larger. Moreover, when the optimal hypothesis is obtained with admissible heuristics, the advantage of island-driven search seems no longer relevant.

However, notice that the optimal solution makes sense only if the input satisfies the constraint that is provided by the language model. In ill-formed utterances that have filled pauses and unknown words, the constraint is violated and searching for an optimal solution is meaningless. When we consider robust understanding of ill-formed utterances, island-driven search is still attractive. This issue is much the same as the comparison of parsing and spotting.

N-best vs. Hypotheses Merging

In continuous speech recognition, some word can be contained in different hypotheses. It is efficient to merge them and avoid redundant matching computation, but it sacrifices the optimality.

Several algorithms merge syntactically or semantically same hypotheses to make full

Table 2.1: List of phone categories

phone category	syllables containing the phone	phone category	syllables containing the phone
p	pa,pu,po	m	ma,mi,mu,mo
py	pja,pi,pju,pe,pjo	my	mja,mju,me,mjo
t	ta,to	n	na,nu,ne,no
ty	ti,te	ny	nja,ni,nju,njo
k	ka,ku,ko	h	ha,he,ho
ky	kja,ki,kju,ke,kjo	hy	hja,hi,hju,hjo
b	ba,bu,bo	f	fu
by	bja,bi,bju,be,bjo	s	sa,su,se,so
d	da,du,do	sh	sja,si,sju,sjo
dy	di,de	z	za,zu,ze,zo
g	ga,gu,go	zy	zja,zi,zju,zjo
gy	gja,gi,gju,ge,gjo	ch	cja,ci,cju,cjo
r	ra,ru,re,ro	ts	cu
ry	rja,ri,rju,rjo		
a		?	?a,?i,?u,?e,?o
i		w	wa
u		ya	ja
e		yu	ju
o		yo	jo

use of beam width or stack size. But this makes it impossible to output the correct N-best hypotheses[29]. The N-best candidates are necessary when the detailed linguistic post-processing can correct or re-order them. To perform a correct N-best search, we must save the N-best candidates at every step of the search without merging them.

2.3 Task and Database

In order to evaluate the recognition methods, we set up several tasks and databases.

2.3.1 Phone Categories

As the acoustic model, we set up 37 phone categories: 27 for consonants, 5 for vowels and 5 for semi-vowels and such. Consonants are classified according to the following vowel contexts. For example, /p/ is classified into 2 categories *p* and *py*, and /pi/ and /pe/ are put into the latter. This classification is done with a clustering technique. Table 2.1 shows the list of the phonemes.

Table 2.2: Specification of databases

task	training	phoneme	word	sentence	ill-formed
speakers	48	48	48	8	8
samples	3119	1317	1317	400	200
utterance	word	word	word	sentence	sentence
class	—	27 consonants	653 words	50 kinds	25 kinds
length			4.7 ph./wd.	6.2 wd./sent.	6.2 wd./sent.

2.3.2 Speech Databases

For training and testing acoustic models, phonetically balanced word database is used. Its vocabulary size is 653, and the set contains all the vowel-consonant-vowel (VCV tuple) patterns in Japanese. The VCV patterns in the word utterances are hand-labeled as to its starting point, its end point and the center of the consonant. The phone models are trained using this VCV patterns, because the transitional parts are important for recognition. The word utterances were made by 48 male speakers. Among all the samples, training samples are chosen so that enough number of samples are provided for each phone. Besides them, testing samples are picked up to evaluate the model.

For the evaluation of the acoustic model itself, phoneme recognition and word recognition is performed. Both are done with the same data mentioned above. Phoneme recognition is on hand-segmented VCV parts. Word recognition is done with vocabulary of 653.

For continuously uttered sentence recognition, we prepared 50 kinds of sample sentences on the 'computer secretary' task domain. They were uttered by 8 male speakers, different from those for word database. We often refer this set to grammatical utterances.

For spotting and robust sentence understanding, we also prepared 25 kinds of ill-formed sentences on the same task domain. These ill-formed sentences have filled pauses or incorrect uses of functional words. They were also uttered by independent 8 male speakers.

Table 2.2 shows the database used for training the phone model and recognition of phonemes, words, and sentences.

Table 2.3: Specification of LR grammars

	G1	G0
vocabulary size	225	210
number of non-terminals	67	52
number of rules	330	317
word perplexity	17.2	37.7

Table 2.4: Specification of syllable bigram

number of syllables in the model	110
size of corpus (in syllables)	119350
number of bigrams that appear in corpus	3671
syllable perplexity with bigram	21.6
syllable perplexity with unigram	46.3

2.3.3 Baseline Language Models

As a baseline language model for continuous speech recognition, we described 2 kinds of LR grammars. Both grammars accept all the grammatical sample sentences but are different in terms of coverage and perplexity. Grammar G0 allows a user more flexible utterances than grammar G1. In grammar G1, the order of phrases such as time and places is fixed, though Japanese expression is relatively flexible. Table 2.3 shows the specification of the grammars. None of the ill-formed sentences are accepted by the both grammars. But by removing filled pauses, many of them could be accepted.

Word-pair grammars are automatically derived from these grammars by checking the possibilities of word connections. The word perplexities of the grammars are about 40 to 50. Word bigram is also derived from the probabilistic version of the LR grammar as a probabilistic word-pair grammar.

In order to estimate syllable (kana) bigram, we used dialogue text database of the Acoustic Society of Japan (ASJ)[30]. It is completely independent from our task domain. The syllable perplexity with bigram gets 21.6. Here we performed back-off smoothing on the bigram of the syllable pairs that do not appear in the corpus frequently[31]. The specification of the bigram is listed in Table 2.4.

Definition of Perplexity

To measure difficulty of recognition task or effect of language modeling, word perplexity is commonly used.

The concept of perplexity is how many choices a recognizer must deal with in each step. When the leaf terminal of the language model is word, word perplexity is defined as how many word candidates can follow the current partial sentence hypothesis on average.

Its formal definition is based on information theory. If a language model is formulated with finite states, entropy at each state is defined using probability $p(w|j)$ that word w appears at state j[32].

$$H(w|j) = -\sum_{w} p(w|j) \log_2 p(w|j) \tag{2.24}$$

When we take the average on all the state, the entropy of language model $H(L)$ is defined.

$$H(L) = \sum_{j} p(j) H(w|j) \tag{2.25}$$

The word perplexity is obtained by taking a power of 2.

$$Q(L) = 2^{H(L)} \tag{2.26}$$

With a language model L_v where only the lexical constraint is available and no grammar is assumed, the word perplexity clearly becomes same as its vocabulary size V.

$$Q(L_v) = 2^{\log_2 V} = V \tag{2.27}$$

With a word-pair grammar L_{wp} where no probabilities are introduced, the word perplexity is as much as the average number of pairs $WP(w)$ per word.

$$Q(L_{wp}) \approx 2^{\sum_w \{\log_2 WP(w)\}/V} \tag{2.28}$$

In a language model such as context-free grammar which cannot be specified with finite states, the entropy or the perplexity is computed by counting the number of possible sentences $W = (w_1, \ldots, w_n)$.

The entropy of the sequence is:

$$H(W) = -\sum_{W} P(W) \log P(W) \tag{2.29}$$

The entropy per word is:

$$\frac{1}{n} H(W) = -\frac{1}{n} \sum_{W} P(W) \log_2 P(W) \tag{2.30}$$

Then, the entropy of the language defined by the context-free grammar is defined as the limit of the per-word entropy as the length of word sequences gets very large[33].

$$H(L_{cfg}) = -\lim_{n\to\infty} \frac{1}{n} \sum_{W} P(W) \log_2 P(W) \tag{2.31}$$

Here, we assume a sequence of sentences delimited with a period. When we assume the equal probability of all possible sequences, the above equation is rewritten with the number of possible sequences.

$$\begin{aligned} H(L_{cfg}) &= \lim_{n\to\infty} \frac{1}{n} \sum^{N_n} \frac{1}{N_n} \log_2 N_n & (2.32) \\ &= \lim_{n\to\infty} \frac{1}{n} \log_2 N_n & (2.33) \\ & N_n : \text{number of sentences of length } n \end{aligned}$$

Actually, we perform the above count by increasing the length n until the value $H(L_{cfg})$ converges. The word perplexity $Q(L_{cfg})$ is computed by equation (2.26). As for the perplexity of a word-pair grammar $Q(L_{wp})$, we adopt the same computation so that it can be directly compared with that of a context-free grammar.

When a more complex language model is incorporated, where it is impossible to count the number of generated sentences, the perplexity has to be estimated with test samples available. This estimated perplexity is referred to test-set perplexity. For a given input word sequence $W = (start, w_1, \ldots, w_n, end)$, it is formulated as follows.

$$Q(testset) = \{\frac{1}{p(w_1|start)} \times \frac{1}{p(w_2|start, w_1)} \times \ldots \times \frac{1}{p(end|start, w_1, \ldots, w_n)}\}^{\frac{1}{(n+1)}} \tag{2.34}$$

If the number of words predicted by the language model for a partial sequence $(start, w_1, w_{m-1})$ is c_m, then the test-set perplexity is computed as follows.

$$Q(testset) = (c_0 \cdot c_1 \cdot c_2 \cdots c_n)^{\frac{1}{(n+1)}} \tag{2.35}$$

The measure is also used for evaluation of context-free grammars for direct comparison.

Chapter 3

Hidden Markov Models (HMM) with Discriminant Analysis

For an improved acoustic model that is the basis of speech recognition, we present discriminative statistical models. Our baseline tools are HMM (Hidden Markov Models) and discriminant analysis. HMM is not only a stochastic model of time-dimensional speech but also a statistical model of the variation of observation patterns. Discriminant analysis is also a statistical method for feature extraction which reduces the parameters and improves the separability between the classes.

Our basic concept is to use discriminant analysis as front-end feature extraction for HMM. We propose a more discriminative model that is based on pair-wise discriminant analyses, since distinctive features vary for different phones and different discriminant functions should be made for different competing classes.

In Section 3.1, we present a classifier based on pair-wise discriminant analyses. In Section 3.2, the classifier is incorporated as a front-end processor of discrete density HMM. In Section 3.3, the pair-wise discriminant analyses are applied to continuous density HMM. In Section 3.4, we compare the discrete model and the continuous model.

3.1 Classifier based on Pair-Wise Discriminant Analyses

3.1.1 Introduction

As the first step of speech recognition, we discuss static pattern recognition. Though speech is a time-dimensional and dynamic pattern, the basis of its recognition is classification or scoring of its local segments frame by frame. It corresponds to the front-end process of HMM, that is conventionally vector quantization or Gaussian distribution modeling of every frame pattern.

We adopt statistical approach, which is general and robust. Linear discriminant analysis or Bayes classifier is effective for extracting valid features of patterns implicitly and for recognizing with stochastic scores. When the number of classes to be discriminated is large, however, it gets difficult to distinguish between close classes due to the use of common variables and a single covariance matrix for all the classes. Therefore the discrimination performance drastically decreases. To overcome this defect, we propose a classifier that is based on pair-wise discriminant analyses, thus we call pair-wise discrimination method. At first, discriminant analyses are performed on the pairs of the classes. Then, by combining the results of these Bayes classifications, the final class is decided.

As the number of classes n increases, there arises a problem that the number of the pairs increases in the order of n^2, which consequently costs too much amount of storage and computation. We also present a method to select pairs that contribute to overall discrimination.

3.1.2 Pair-Wise Discrimination Method

Linear Discriminant Analysis

Firstly, we formulate the conventional discriminant analysis in a short form[34].

The input to a classifier is a d-dimensional pattern vector $\mathbf{x} = [x_1, x_2.., x_d]$. Suppose the population of class c_i is normally distributed with mean μ_i and covariance Σ_i, and suppose further covariance matrices $\Sigma_i (i = 1, .., n)$ are equal to Σ. The probability density function that a given pattern $\mathbf{x}$ belongs to class c_i is as follows.

$$\begin{aligned} p(\mathbf{x}|c_i) &= C \cdot exp\{-D_i^2(\mathbf{x})/2\} \quad (3.1) \\ D_i^2(\mathbf{x}) &= (\mathbf{x} - \mu_i)^t \cdot \Sigma^{-1} \cdot (\mathbf{x} - \mu_i) \quad (3.2) \\ C &= (2\pi)^{-d/2} \cdot |\Sigma|^{-1/2} \quad (3.3) \end{aligned}$$

Here $D_i^2(\mathbf{x})$ is called Mahalanobis distance between a given pattern $\mathbf{x}$ and mean $\mathbf{u}_i$ of class c_i. When a priori probability of every class is unknown or equal, a pattern $\mathbf{x}$ is classified into the class whose probability density function $p(\mathbf{x}|c_i)$ is maximum, namely the class to which Mahalanobis distance $D_i^2(\mathbf{x})$ is minimum. This classification mechanism is called Bayes classier. The discriminant function f_i gets linear when we assume covariance matrices of the classes to be equal[15].

$$\begin{aligned} f_i(\mathbf{x}) &= \mathbf{x}^t \cdot \Sigma^{-1} \cdot \mu_i - \frac{1}{2}\mu_i^t \cdot \Sigma^{-1} \cdot \mu_i & (3.4) \\ &= \sum_{k=1}^{d} a_k \cdot x_k + a_0 & (3.5) \end{aligned}$$

In discriminant analysis to construct a classifier, all the input variables x_i are not necessarily relevant to discrimination. Some variables may be useless or some can be substituted by others. Therefore variables that separate the classes, namely those which maximize Mahalanobis distance, should be selected. This variable selection uses F-value, a measure for improvement of Mahalanobis distance by adding or removing a variable. It is performed step-wise until no addition nor removal of a variable occurs[35]. As a result, the dimension of $\mathbf{x}$ is reduced.

In two-class pattern discrimination, optimal variables to separate them are selected. As the number of classes increases, however, it is difficult to select variables to maximize Mahalanobis distance among all the classes. Every variable is effective to separate some classes, but it may be useless or even harmful to separation of others. For example, it is evident that the effective features to distinguish phoneme /b/ and /m/ is different from those for /b/ and /d/. In the conventional one-stage discriminant analysis, variables that contribute to overall discrimination on the average are selected. Consequently the variables that feature a class are not selected if they have no discriminating power for other classes. Therefore, different discriminant function should be adopted for different classes.

Combining Pair-Wise Classifications

To overcome the defect described above, we have proposed pair-wise discrimination method[36], that combines Bayes classifiers of the pairs of the classes. This method is based on the property that, in two-class discrimination, an optimal discriminant function is constructed to separate them.

At first, a set of pairs of the classes is constructed. The number of the pairs can be its possible combination ${}_n\mathsf{C}_2$ at most, where n is the number of the classes, but we select the pairs that are effective for overall discrimination. Details of how to select pairs are discussed later. Next, for each pair of the classes, statistical variable selection is performed to make a discriminant space that maximizes Mahalanobis distance between

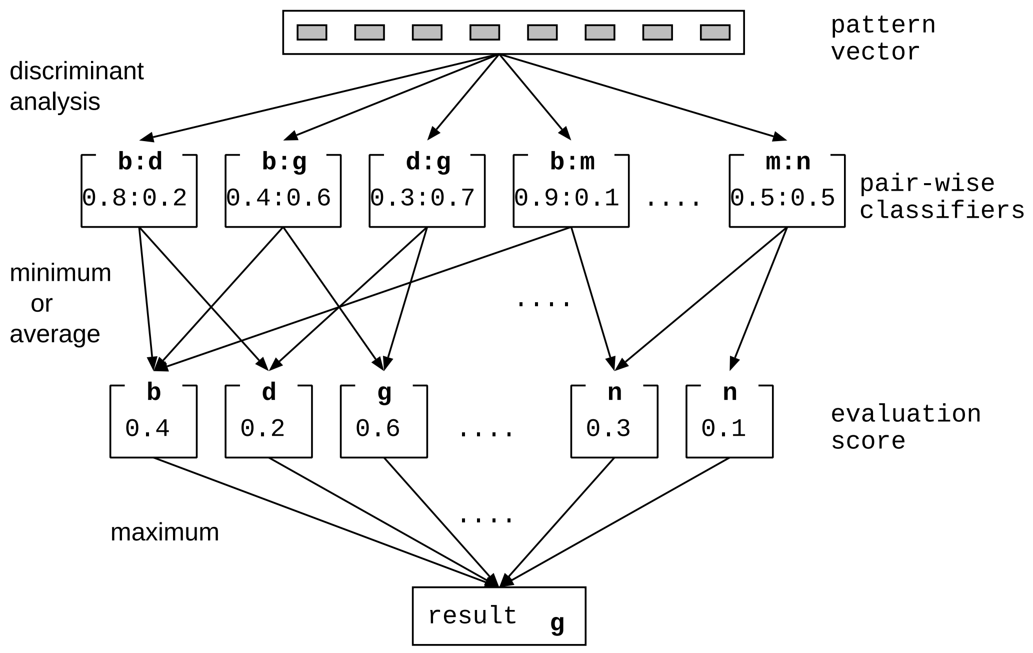

Figure 3.1: Pair-wise discrimination method

the two. Here, we estimate a specific covariance matrix for each pair, which reduces the deviation compared with the discriminant analysis using a common covariance matrix for all the classes, while maintaining linear discriminant function.

For a given pattern, the two-class discriminations are done with the dedicated Bayes classifiers. Then, by combining the results of the pair-wise classifications, the evaluation score for each class is obtained, by which the pattern is classified finally. These steps are illustrated in Fig 3.1.

3.1.3 Combining Pair-Wise Classifications

We present two methods to combine the pair-wise classifications: minimax method and majority method.

(1) Minimax method

The method uses a posteriori probability $p_{i-j}(c_i|\mathbf{x})$ that is obtained on the pair classifier for c_i and class c_j by the following Bayesian rule, assuming even a priori probabilities.

$$p_{i-j}(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)}{p(\mathbf{x}|c_i) + p(\mathbf{x}|c_j)} \tag{3.6}$$

Even though pattern $\mathbf{x}$ belongs to neither class c_i nor class c_j, either $p_{i-j}(c_i|\mathbf{x})$ or $p_{i-j}(c_j|\mathbf{x})$ will be more than 0.5. Therefore, we cannot say the pattern $\mathbf{x}$ comes from class

c_i if $p_{i-j}(c_i|\mathbf{x})$ is large. We just conclude that it does not if $p_{i-j}(c_i|\mathbf{x})$ is small.

The evaluation function $f_i(\mathbf{x})$ for class c_i is defined as the minimum of a posteriori probabilities on the pairs that include class c_i.

$$f_i(\mathbf{x}) = \min_j \; p_{i-j}(c_i|\mathbf{x}) \tag{3.7}$$

This parameter reflects to what extent the class is not denied by pair-wise classifications. A given pattern is classified where the evaluation score is the largest, namely the class that is not denied by most pairs.

Actually, a posteriori probability can be computed quite fast with the two-class linear discriminant function. The ratio of the probability density is computed with the linear discriminant function g_i^{i-j}.

$$\begin{aligned}
\frac{p_{i-j}(\mathbf{x}|c_i)}{p_{i-j}(\mathbf{x}|c_j)} &= \frac{C \cdot \exp\{-\frac{1}{2}(\mathbf{x}-\mu_i)^t \cdot \Sigma^{-1} \cdot (\mathbf{x}-\mu_i)\}}{C \cdot \exp\{-\frac{1}{2}(\mathbf{x}-\mu_j)^t \cdot \Sigma^{-1} \cdot (\mathbf{x}-\mu_j)\}} && (3.8)\\
&= \frac{\exp\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x} + \mathbf{x}^t\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i\Sigma^{-1}\mu_i\}}{\exp\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x} + \mathbf{x}^t\Sigma^{-1}\mu_j - \frac{1}{2}\mu_j\Sigma^{-1}\mu_j\}} && (3.9)\\
&= \frac{\exp\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x}\} \cdot \exp\{\mathbf{x}^t\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i\Sigma^{-1}\mu_i\}}{\exp\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x}\} \cdot \exp\{\mathbf{x}^t\Sigma^{-1}\mu_j - \frac{1}{2}\mu_j\Sigma^{-1}\mu_j\}} && (3.10)\\
&= \frac{\exp\{g_i^{i-j}(\mathbf{x})\}}{\exp\{g_j^{i-j}(\mathbf{x})\}} && (3.11)
\end{aligned}$$

A posteriori probability is obtained with this ratio, consequently in the linear order.

$$\begin{aligned}
p_{i-j}(c_i|\mathbf{x}) &= \frac{p_{i-j}(\mathbf{x}|c_i)}{p_{i-j}(\mathbf{x}|c_i) + p_{i-j}(\mathbf{x}|c_j)} && (3.12)\\
&= \frac{1}{1 + \frac{p_{i-j}(\mathbf{x}|c_j)}{p_{i-j}(\mathbf{x}|c_i)}} && (3.13)
\end{aligned}$$

(2) Majority method

The method uses a binary value $q_{i-j}(c_i|\mathbf{x})$ that is obtained with the linear discriminant function $g_i^{i-j}(\mathbf{x})$.

$$q_{i-j}(c_i|\mathbf{x}) = \begin{cases} 1 & g_i^{i-j}(\mathbf{x}) > g_j^{i-j}(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

It is nothing but transforming a posteriori probability $p_{i-j}(c_i|\mathbf{x})$ into a binary value.

$$q_{i-j}(c_i|\mathbf{x}) = 1 \Leftrightarrow p_{i-j}(c_i|\mathbf{x}) > 0.5$$

Using this value further saves the amount of computation by obtaining the differential discriminant function $\{g_i^{i-j}(\mathbf{x}) - g_j^{i-j}(\mathbf{x})\}$ beforehand.

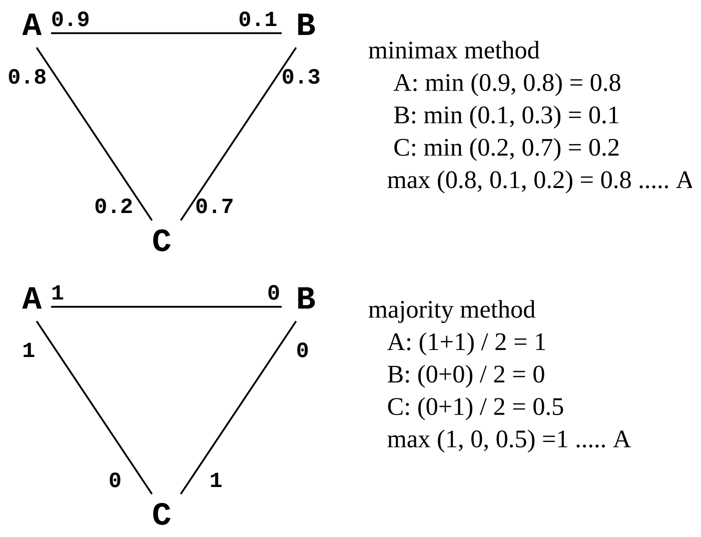

Figure 3.2: Examples of minimax method and majority method

The evaluation function $f_i(\mathbf{x})$ for class c_i is obtained by taking the average of the binary values on the pairs that include class c_i.

$$f_i(\mathbf{x}) = \text{average } q_{i-j}(c_i|\mathbf{x}) \tag{3.15}$$

This parameter reflects the ratio of the pairs supporting the class. A given pattern is classified where the evaluation score is the largest, namely the class that is supported by most pairs. Majority method is equivalent to minimax method if and only if just one class is supported by all the associated pairs.

Examples of these methods applied to three-class {A, B, C} discrimination are shown in Fig. 3.2

3.1.4 Selecting Effective Pairs

Pair-wise discrimination method needs much computation and storage if we use all the possible pairs since the number of the pairs increases in the order of n^2 (n: number of classes). We, therefore, present an algorithm to select pairs that are effective for overall discrimination.

To find out effective pairs, we introduce some measures to indicate to what extent each pair contributes to overall discrimination.

(1) Occurrence of pair $[c_i - c_j]$ as the first and the second candidate: α_{i-j}

For all the samples, we count the occurrence of the first candidate and the second candidate pairs since these pairs are likely to be confused and are essential in pair-wise discrimination method. In other words, without this pair, pair-wise discrimination method would fail for the sample.

(2) Change of the accuracy according to addition / removal of pair $[c_i - c_j]$: β_{i-j}

This is a direct measure for the performance of discrimination. We define two measures: one corresponds to addition and the other to removal.

$$\begin{aligned}\beta^-_{i-j} = \max_k \ & \{(\text{accuracy of class } c_k \text{ with present pairs including } [c_i - c_j]) \\ & -(\text{accuracy of class } c_k \text{ if the pair removed})\}\end{aligned}$$

$$\begin{aligned}\beta^+_{i-j} = \max_k \ & \{(\text{accuracy of class } c_k \text{ with present pairs excluding } [c_i - c_j]) \\ & -(\text{accuracy of class } c_k \text{ if the pair added})\}\end{aligned}$$

Use of the measure β_{i-j} needs much computation because all the samples must be tested to obtain each β_{i-j} and calculation is repeated until the set of pairs converges. Therefore, measure α_{i-j}, which needs only one test for all the samples, is preferable. But only with this measure, we may overlook some important pairs because it does not directly reflect the recognition performance. So we use measure α_{i-j} as the primary and measure β_{i-j} as the complementary.

Based on the those measures, an algorithm to select effective pairs is realized. The basic strategy is to eliminate unnecessary pairs by using measure α_{i-j} and recover the pairs that are once eliminated but are necessary by using measure β^+_{i-j}. However, there may be pairs that can be substituted by others even if they remain after the first step. Such pairs are found out among those whose α_{i-j} are low by using measure β^-_{i-j} and eliminated in the second step. The algorithm is as follows.

1. With all the possible pairs, pair-wise discrimination method is performed for all the samples and α_{i-j} for each pair is obtained.

   If $\alpha_{i-j} = 0$, then remove pair $[c_i - c_j]$.

2. For those pairs whose α_{i-j} are lower than θ_a, β^-_{i-j} are calculated.

   If $\beta^-_{i-j} < \theta_{b1}$, then remove pair $[c_i - c_j]$.

Table 3.1: Result of static phoneme segment recognition

method	accuracy (%)
conventional one-stage discriminant analysis	78.3
pair-wise discrimination method (minimax)	93.1
pair-wise discrimination method (majority)	92.2

3. For those pairs which are removed and whose confusions increase by θ_{b2} compared with the case with all the possible pairs, β^{+}_{i-j} are calculated.

   If $\beta^{+}_{i-j} > \theta_{b3}$, then add pair $[c_i - c_j]$.

3.1.5 Experimental Evaluation

We have evaluated pair-wise discrimination method and the algorithm to select effective pairs on the phoneme recognition. The phonemes to be discriminated are all the Japanese consonants except *f* in Table 2.1. The number of the classes is 26. The way of experiments here is somewhat different from those in other sections. First, the central frames of the consonants in all the samples are segmented manually. The input of classification is the static pattern of 8 frames around the central frame (preceding 2 and following 5 frames). Second, all the 3696 samples are used for both training and test data. In order to realize open recognition experiments, we perform Jack-Knife discrimination[37] which, in recognizing a given sample, uses discriminant function calculated by the samples excluding that one.

At first, we applied the conventional one-stage discriminant analysis pair-wise discrimination method that uses all the possible pairs of the phones. The results are shown in Table 3.1. The conventional multiple-class discriminant analysis is not effective for a large number of classes (26 in this case). There was a lot of confusion between similar phonemes, for example, [*ty*-*ky*] and [*s*-*ts*]. Pair-wise discrimination method reduced recognition errors to half of those by the conventional method. Majority method got a bit worse recognition rate than minimax method. It shows the use of binary results of the pair-wise classifications sacrifices the accuracy to some extent. But the accuracy decrease is negligible compared with the conventional Bayes classifiers.

Then, we tried to reduce the pairs using the selection algorithm. The thresholds θ_a, θ_{b1}, θ_{b2}, and θ_{b3} are set to 30, 0.1%, 0.1%, 0.1%, respectively. As for minimax method, the number of the remaining pairs and the recognition rate at each step of the algorithm

Table 3.2: Reduction of pairs by selection algorithm

	number of pairs	accuracy (%)
initial	325	94.8
step 1	121	94.6
step 2	77	93.9
step 3	79	94.2

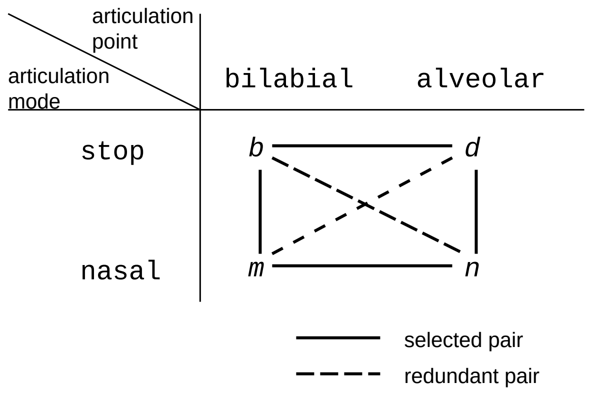

Figure 3.3: An example of redundant pairs

are shown in Table 3.2. We could reduce the pairs to 79, which is about one fourth of all the possible pairs, with only 0.6% accuracy decrease. This means that there is much redundancy in the set of pair-wise classifiers. The rates shown here are based on closed experiments, and the rate of an open experiment with 79 pairs was 92.5%.

Now we discuss where the redundancy comes from. From the viewpoint of pattern recognition, pair-wise discrimination method needs only pairs of the classes that are closely located in the pattern space and the results of the classifications of the eliminated pairs can be inferred from the results of the remainders. It is proved by the fact that most of the pairs are eliminated in the first step, namely by measure α_{i-j}. From the viewpoint of phonetics, these closely located pairs of phonemes have the same manners of articulation or the same points of articulation. An example is shown in Figure 3.3.

Another experiment showed that majority method needed 115 pairs, which were more than those for minimax method and comparable to the number at step 1 of Table 3.1. Minimax method uses a probabilistic measure to determine which class is more probable and can eliminate some pairs of the closely located classes by using measure β_{i-j}. But this is not possible for majority method. Table 3.3 shows, in majority method, how many classes are not denied by any classifiers. When more than two classes are supported by all

Table 3.3: Content of discrimination by majority method

(with 115 pairs)

number of classes not denied by any pairs	correct	error
more than 2	0.0%	0.4%
1	93.3%	4.6%
0	0.7%	1.0%

the associated pairs, discrimination clearly fails. With 115 pairs, these cases seldom occur. The ratio of the cases where no classes were supported by all the associated pairs was also small. This fact shows that, in most cases, the classifier of some pair, which is consequently of the first candidate and the second candidate, directly affects overall discrimination. Therefore, pair-wise discrimination method that uses optimal two-class classifiers does not lower the recognition performance even if the number of classes increases.

3.1.6 Conclusion

We have pointed out the defect of the conventional one-stage multiple-class discriminant analysis and proposed pair-wise discrimination method that combines optimal Bayes classifiers of the pairs of the classes. We have further proposed an algorithm to select effective pairs to reduce necessary computation and storage. Experimental results show that pair-wise discrimination method is much more effective and it is realized with only the pairs of the closely located classes.

3.2 Discrete HMM with Classifier Front-End

3.2.1 Introduction

As the basis of a whole speech recognition system, we discuss the acoustic model. It is based on the classification of its local segments combined with dynamic time-warping. HMM (Hidden Markov Model) is a powerful method to perform dynamic time-warping of speech and is easily extended to recognize larger units such as words or sentences. It, however, does not have enough power to discriminate competing classes as it is trained with in-class data only. Moreover, VQ (Vector Quantization)[38] that is conventionally used for discrete HMM just performs quantization of input patterns, not its discrimination.

Our approach is classifier-based HMM, which is HMM with a classifier as a front-end processor instead of VQ. The classifier is trained to separate input patterns of different phones so that HMM can easily distinguish them.

We apply pair-wise discrimination method for the classifier. Since it is based on statistical discriminant analysis, it is straightforward to combine it with HMM that is also a stochastic modeling. Furthermore, we combine the score of the classifier on the local features and the score of HMM on the global features to get a recognition result. How to combine those two scores is also discussed.

3.2.2 Pair-Wise Discriminant Discrete HMM (PWD-D-HMM)

HMM realizes dynamic time-warping with its probabilistic transition of the states and copes with the variation of observation sequences with its probabilistic distributions.

In conventional discrete HMM[10], an input sequence consists of codes determined by VQ. VQ reduces the amount of computation and storage, but quantization error is inevitable. This error is the loss of information contained in the pattern vectors and hampers recognition performance.

Furthermore, the training algorithm commonly used lacks a concept of maximizing the distance between the classes. Typically, a phone HMM is trained with samples that belongs to the class independently. Every model is optimal for the samples, but it is not trained to discriminate from other classes.

For those reasons, discriminating power of HMM is not sufficient especially to distinguish acoustically similar phones.

On the other hand, discriminant analysis is effective to analyze local patterns of phones such as a burst or a friction, though it does not perform dynamic time-warping. It extracts effective features to separate competing classes statistically.

In Table 3.4 the two stochastic methods described above are compared. As we can see, discriminant analysis and HMM are different in feature extraction, and they can complement or cooperate each other. Here we propose a combined method.

In order to improve the discriminating ability of discrete HMM, it is significant to separate observation symbol sequences of the different phones so that HMM can easily distinguish them. Therefore, a phonetic element classifier based on discriminant analysis is constructed. Each phonetic element belongs to one phone and represents a local pattern of the phone, for example, a burst or a friction. Some phonetic elements such as a murmur

Table 3.4: Comparison between discriminant analysis and HMM

	discriminant analysis	HMM
input	several frames	dozens of frames
	vector	Markov sequence
feature extraction	local	global
	combinatorial	dynamic
merit	discriminating power	time-warping

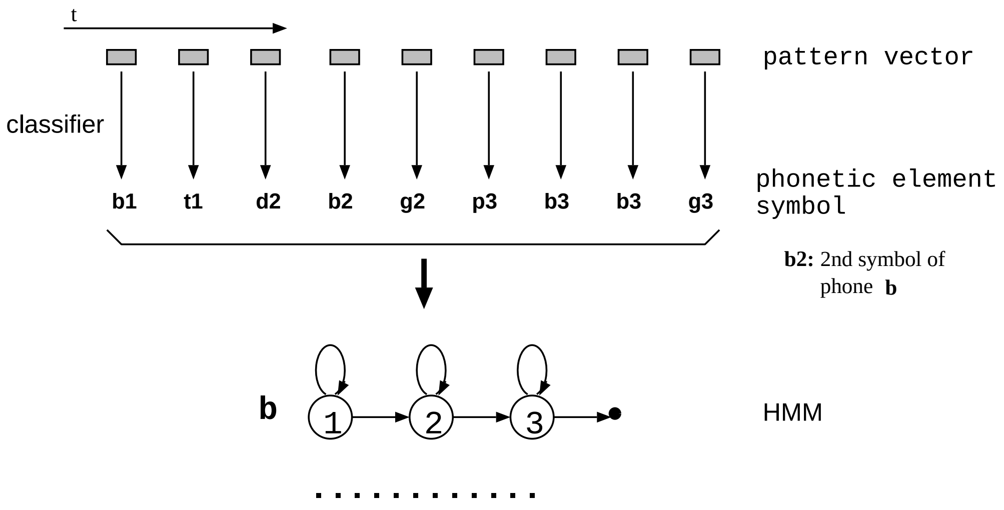

Figure 3.4: Frame labeling for discrete HMM with classifier front-end

part are common to multiple phones. Those phonetic element parts are initially specified by human observation and input to discriminant analysis to construct a Bayes classifier.

Then, HMM with the phonetic element classifier front-end is constructed. The classifier scans an input speech and outputs a phonetic element symbol sequence, which is processed by HMM. This procedure is illustrated in Figure 3.4. Here, adjacent frames are used to make frame labeling to reflect dynamic features. It is nothing but a replacement of VQ on conventional discrete HMM with the classifier. The difference is that the symbols here explicitly belong to some phone and the classifier is trained to discriminate them.

Since the classifier extracts optimal features to separate the phonetic elements, the obtained symbol sequence for each phone gets easily distinguishable. As a consequence, separability of phone HMMs will be improved. From the other viewpoint, when we regard HMM as a post-processor, HMM performs dynamic time-warping on the results of the classifier.

Classification of the local phonetic elements is done by pair-wise discrimination method,

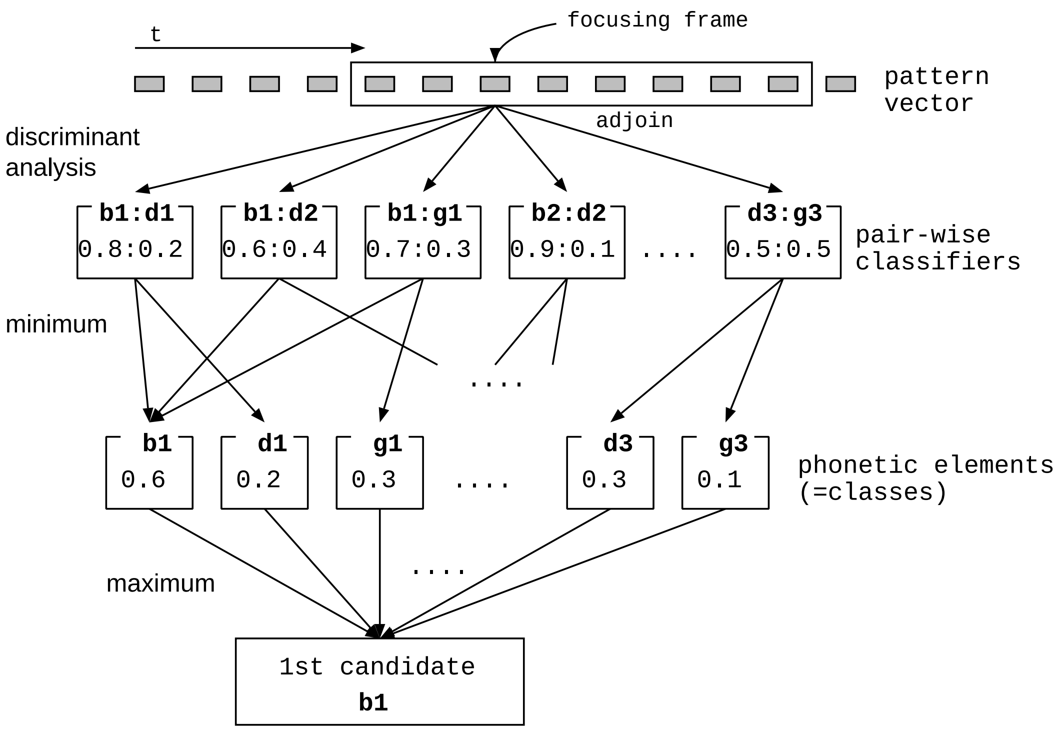

Figure 3.5: Pair-wise Bayes classifiers for frame labeling

a combination of Bayes classifiers. Its discriminant function for class o_i is formulated in the previous section as follows, though the classifier here discriminates phonetic element symbols, of which phones consist. It is illustrated in Figure 3.5, where *b1* represents the first phonetic element of phone *b*.

$$f_i = \min_j \; p_{[i-j]}(o_i|\mathbf{x}) \tag{3.16}$$

$p_{[i-j]}(o_i|\mathbf{x})$: probability of o_i by $o_i - o_j$ classification

Thus, we call the whole recognition mechanism Pair-Wise Discriminant Discrete HMM (PWD-D-HMM). As similar approaches, LVQ-HMM[39], TDNN-HMM[40] and so on have been proposed. We use a statistical method as front-end, thus it gets possible to combine the score of the classifier with HMM both theoretically and practically.

Combining Scores

In the prototypical method, a sequence of the most probable symbols at the frames of speech is obtained by the statistical classifier, and Forward or Viterbi score is computed for each phone HMM to make recognition. We further propose to combine the scores of the two stochastic methods. The score of Bayes classifier reflects the local features of phones and that of HMM reflects the global features. By combining them, therefore, we expect more precise recognition that reflects the merits of both methods.

Three methods to combine these scores are presented.

(1) Fuzzy Frame Labeling

The deterministic frame labeling would be harmful because the frame-wise discrimination is not free from classification errors. To solve the problem, we can use fuzzy labeling that passes multiple candidates at each frame to HMM. This is analogous to fuzzy vector quantization (FVQ)[41], but the weight $p(o_{(j)}|\mathbf{x})$ to the symbol output probability $p(o_{(j)}|s_i)$ is defined formally as the probability computed by the classifier. Actually, it is similar to semi-continuous HMM[42] or tied-mixture continuous HMM[43] when we use $p(\mathbf{x}|o_{(j)})$ instead of $p(o_{(j)}|\mathbf{x})$. The symbol output probability for HMM state s_i is modified as follows.

$$p(o_{(1)}|s_i) \rightarrow \sum_j p(o_{(j)}|\mathbf{x}) \cdot p(o_{(j)}|s_i) \tag{3.17}$$

$o_{(j)}$: j-th best candidate symbol at the time-frame

$\mathbf{x}$: pattern vector at the time-frame

(2) Precise Classification after Viterbi Segmentation

The method is to incorporate the score of Bayes classifier after Viterbi alignment. The scores of the phonetic element classifier are averaged for each phone which the phonetic elements belong to, where the phone segment is aligned by Viterbi algorithm. Finally, it is combined with the HMM score. The similar method is HMM/LVQ[44].

For a given input pattern sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ and its classified symbol sequence $O = (o_1, \ldots, o_T)$, the score of phone model m is computed as follows.

1. Perform Viterbi algorithm to get the score of HMM $p(O|m)$.
2. Perform Viterbi segmentation of the input by back-tracing the most probable states.
3. Compute the average score of Bayes classifier for the aligned phone segments.
   $p_D(m|\mathbf{X}) = \frac{1}{T} \sum_{o_t \in m} p(o_t|\mathbf{x}_t)$
4. Combine the HMM score and the classifier score.

$$p_M(O|m) \times p_D(m|\mathbf{X}) \tag{3.18}$$

(3) Frame-Level Combination

The method is to combine the score of Bayes classifier with the score of HMM at the speech frame level, namely to integrate the score of the classifier into forward probability computation of HMM.

This is done by modifying the output probability $p(o_{(1)}|s_i)$ of phone HMM m so that the scores of phonetic elements of m are reflected.

$$p(o_{(1)}|s_i) \rightarrow p(o_{(1)}|s_i) * \sum_j p(o_j^m|\mathbf{x}) \tag{3.19}$$

$o_{(1)}$: the best candidate symbol at the time-frame

o_j^m: j-th phonetic element symbol belonging to phone m

The factor $p(o_j^m|\mathbf{x})$ is the probability by Bayes classifier that the frame pattern represents the local feature of the phone m.

3.2.3 Experimental Evaluation

We have evaluated discrete HMM based on pair-wise discriminant analyses (PWD-D-HMM). The task here is to recognize 27 phonemes (consonant) in word utterances. We define 127 phonetic elements to represent a variety of local patterns. The number of HMM states is 3 or 4.

We have realized prototypical PWD-D-HMM, then three methods to combine the score of the classifier. We have also performed VQ-based HMM for comparison. As for VQ-HMM, we have tested two types. One uses single codebook of size 256, and the other uses separate codebooks for spectrum and power[11], with 256 and 64 entries respectively.

The recognition results are listed in Table 3.5. PWD-D-HMM achieves higher accuracy than conventional VQ-HMM. Although VQ-HMM with multiple codebook gets a comparable recognition rate for closed samples, it lowers the accuracy for open data. It fails to cope with the variation of input speech of many speakers. Actually, with VQ-HMM, there was a lot of confusion between acoustically similar phones, for example, $p \rightarrow t$ and $m \rightarrow n$. PWD-D-HMM succeeds in extracting features independent of speakers and contexts.

Combining the scores had some effect but not so much. There was no difference between the score combination methods. But careful analysis is necessary by performing word or sentence recognition.

Table 3.5: Phoneme recognition result with discrete HMM

recognition method	closed	open
PWD-D-HMM (normal)	93.1	82.4
PWD-D-HMM (fuzzy frame labeling)	89.5	83.6
PWD-D-HMM (Viterbi/classification)	92.6	83.9
PWD-D-HMM (frame-level combination)†	93.4	84.0
VQ-HMM (single codebook)	81.1	61.7
VQ-HMM (multiple codebook)	89.4	67.7

† used samples are slightly different

3.2.4 Conclusion

We have proposed a new discriminative acoustic model named PWD-D-HMM based on two stochastic methods: discriminant analysis and HMM. Pair-wise Bayes classifiers precisely discriminate the local patterns of phones called phonetic elements. HMM grasps the global patterns and performs dynamic time-warping. Experimental results show that our combined method is more effective than conventional VQ-HMM. We have also proposed to combine the scores of the two methods and confirmed its effectiveness.

3.3 Continuous HMM with Pair-Wise Discriminant Analyses

3.3.1 Introduction

We have studied the acoustic model to improve its discriminant ability by incorporating a classifier front-end. Another requirement is that real value output of the acoustic model is desirable. Moreover, the output value should be really stochastic to realize flexible interface with the natural language processing unit. Although most of the discriminative training strategies have achieved high performance in phoneme recognition, their scores are almost digitized and inadequate for recognizing larger units, such as words or sentences[45].

In this section, we propose a new architecture of continuous density HMM which is based on pair-wise discriminant analyses to realize both discriminant ability and stochastic scoring.

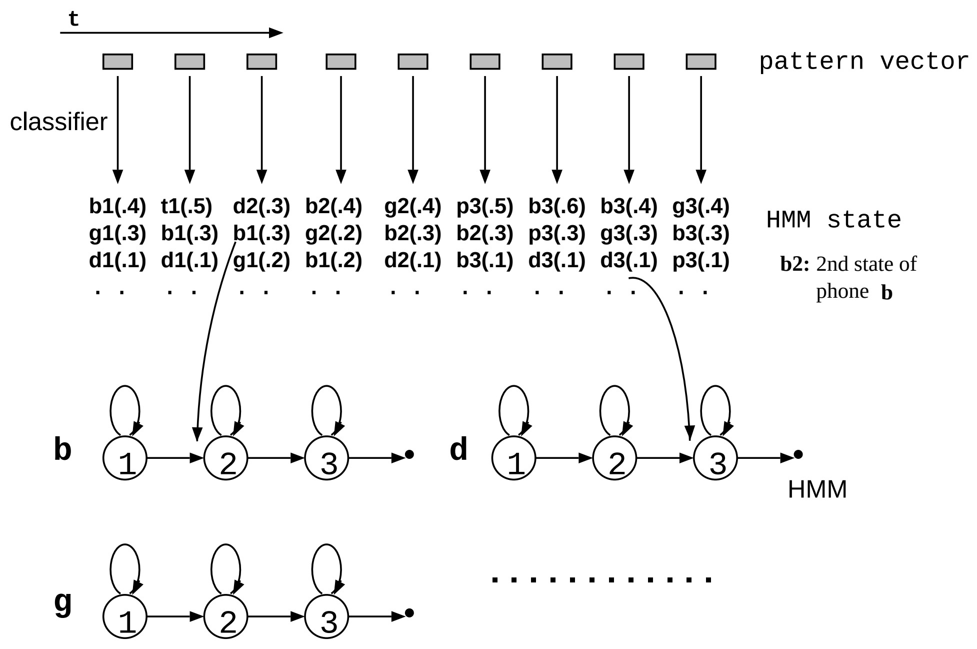
t
pattern vector
classifier
b1(.4) t1(.5) d2(.3) b2(.4) g2(.4) p3(.5) b3(.6) b3(.4) g3(.4)
g1(.3) b1(.3) b1(.3) g2(.2) b2(.3) b2(.3) p3(.3) g3(.3) b3(.3)
d1(.1) d1(.1) g1(.2) b1(.2) d2(.1) b3(.1) d3(.1) d3(.1) p3(.1)
HMM state
b2: 2nd state of phone b
b
d
g
1
2
3
HMM

Figure 3.6: Computing output probabilities of HMM states with classifier

3.3.2 Pair-Wise Discriminant Continuous HMM (PWD-C-HMM)

As a more elaborate model that avoids deterministic frame labeling or quantization, continuous distribution HMM[46] have been studied. In order to make continuous HMM to be more discriminative, it is significant to improve separability of the states of the models. Thus, the problem is reduced to constructing an optimal classifier that discriminates the HMM states, when we regard the HMM states as classes. Output probabilities of the HMM states are computed by the classifier as its scores. In other words, we incorporate a classifier into continuous HMM to compute output probabilities through discriminating mechanism instead of simple Gaussian distribution. This procedure is illustrated in Figure 3.6. Here, adjacent frames are used to make frame-wise scoring to reflect dynamic features.

The classifier we adopt is Bayes classifier. Another implementation is to use emissions of a neural network as output probabilities[47],[48]. Since Bayes classifier is based on statistical decision theory, it can be naturally integrated with HMM, and guarantees the stochastic distribution of the output values. The correspondence between Bayes classifier and HMM is listed in Table 3.6.

As a statistical method, use of simple discriminant analysis or subspace projection that improves separability between in-class data and out-of-class data is proposed[21],[49],[50].

Table 3.6: Correspondence between Bayes classifier and HMM

Bayes classifier	HMM
class multiple template	state mixture density
average covariance	average covariance
—	state transition probability

They perform a uniform vector transformation for all the competing states of all the models. However, discriminant features vary for different states or different models. For example, effective features to distinguish phoneme /p/ and /t/ completely differ from those to separate /p/ and /m/. Therefore, different feature extraction should be made for each of them.

Here we propose to construct a classifier with optimal vector transformation for every pair of the states, and compute the output probabilities of the HMM states through the set of pair-wise Bayes classifiers. For example, a classifier for s_i^m and s_j^n (s_i^m: i-th state of model m) sets an optimal pattern space to separate s_i^m and s_j^n. Since each classifier has a different pattern space, it just computes a relative value of the probabilities of the two states, and actual output probabilities are computed by combining and normalizing the relative values. Training of the classifiers are iterated and interacted with that of HMM, so that the classifiers and HMM are fully integrated and totally optimized. We name this recognition mechanism Pair-Wise Discriminant Continuous HMM (PWD-C-HMM).

3.3.3 Recognition and Training Algorithm

Recognition Algorithm

Recognition algorithm of PWD-C-HMM is the same as that of conventional continuous density HMM, except that output probabilities $p(\mathbf{x}|s_i^m)$ (s_i^m:state, $\mathbf{x}$:pattern vector) are computed by the following steps. These steps are illustrated in Figure 3.7.

1. Pair-wise classifiers are applied on a given pattern $\mathbf{x}$. For every pair of the states s_i^m-$s_j^n(m{\neq}n)$, a classifier ranks the two states, and computes a relative value of the probabilities $\{p_{[mi-nj]}(\mathbf{x}|s_i^m) : p_{[mi-nj]}(\mathbf{x}|s_j^n)\}$, using the dedicated discriminant space for $[s_i^m$-$s_j^n]$ classification. The probabilities are transformed to a priori form.

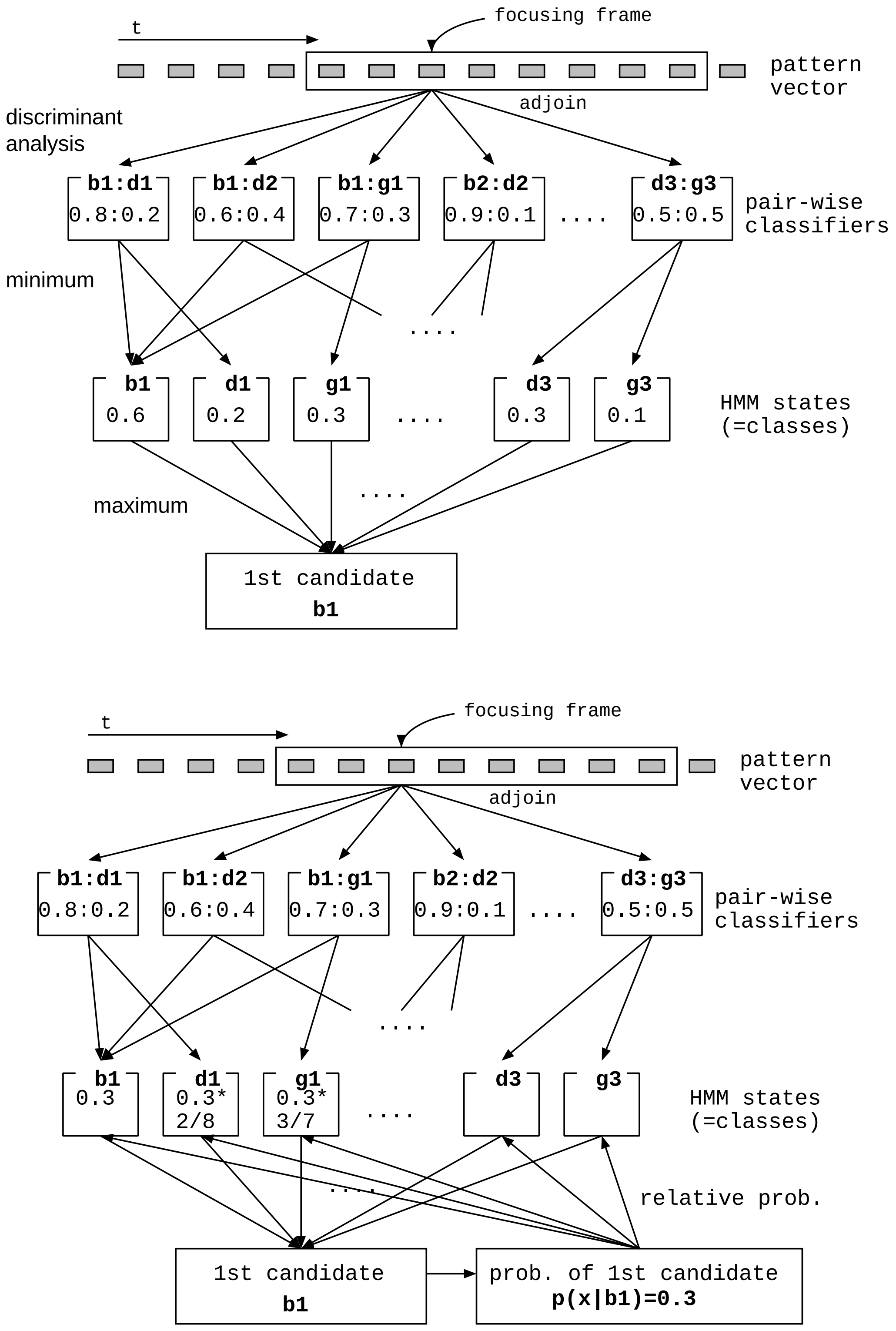

Figure 3.7: Pair-wise Bayes classifiers for output probability computation

2. The most probable class $s_{(1)}$ is picked up by combining the results of the pair-wise classifications. We adopt minimax method, as exemplified in Figure 3.2, which chooses the class winning the highest value of the following discriminant function (see the upper part of Figure 3.7).

$$f_{(mi)}(\mathbf{x}) = \min_{n,j} \; p_{[mi-nj]}(s_i^m|\mathbf{x}) \tag{3.20}$$

3. Output probabilities of the HMM states are computed by normalizing the relative values of the pair-wise classifications. We first calculate a probability for the previously chosen first candidate $p(\mathbf{x}|s_{(1)})$ under a canonical Gaussian distribution, then calculate probabilities for the other states $p(\mathbf{x}|s_i^m)$ as its relative values (see the lower part of Figure 3.7).

$$p(\mathbf{x}|s_{(1)}) * \frac{p_{[(1)-mi]}(\mathbf{x}|s_i^m)}{p_{[(1)-mi]}(\mathbf{x}|s_{(1)})} \tag{3.21}$$

In the example of Figure 3.7, state *b1* is chosen as its minimum of a posterior probabilities on associated pairs is maximum, and its probability 0.3 is computed with a canonical distribution, then the probabilities of other states are computed as the relative values against that of state *b1*.

In practice, all the possible pairs of the states are not necessary. We only have to set up those pairs which are likely to be confused[51]. For a state whose pair with the first candidate is omitted, a small constant ϵ is assigned to its output probability. We also set a floor of lower probabilities to the value ϵ.

Both the relative value of probabilities and a posteriori probability are computed in the linear order when we use a linear discriminant function for each pair-wise classification, as proven in Section 3.1. The canonical distribution as in conventional continuous HMM is used to normalize the probabilities at different frames.

Training Algorithm

HMM is usually trained by Forward-Backward algorithm. In PWD-C-HMM, training of the pair-wise classifiers are integrated with it, as an iteration of the following steps.

1. For every HMM, state transition probabilities are estimated by Forward-Backward algorithm.

2. For all the training samples, Viterbi alignment is performed to assign each frame pattern vector to some state of HMM.

3. For every pair of the states, feature extraction based on discriminant analysis is performed to construct a Bayes classifier, using the pattern vectors assigned in the previous step.

In Forward-Backward and Viterbi computation, output probabilities of the HMM states are calculated through pair-wise classifiers. At the initial step, however, we can use conventional computation.

In Step 2, we perform Viterbi alignment to get pattern vectors explicitly. As a more sophisticated model, it is possible to directly use the results of Forward-Backward computation of Step 1 for discriminant analysis of Step 3. To be more concrete, the means and the covariances computed by Forward-Backward algorithm, which is formulated in equation (2.18) and (2.19), can be provided for discriminant analysis as they are, by omitting Step 2.

Probability Normalization

As the form of the output probability, the followings are compared.

(1) probability density

A value of probability density function of Gaussian distribution is normally used in continuous HMM.

$$\begin{aligned} p(\mathbf{x}|s_i^m) &= C \cdot \exp(-D_{mi}^2/2) \qquad (3.22) \\ D_{mi}^2 &= (\mathbf{x} - \mu_{mi})^t \cdot \Sigma^{-1} \cdot (\mathbf{x} - \mu_{mi}) \\ & \quad \mu_{mi} : \text{mean}, \Sigma : \text{covariance} \end{aligned}$$

(2) a posteriori probability

This is a normalization of probability density by its sum over all the classes. In our method, it is computed in the linear order without probability density calculation. Neural networks theoretically output this form.

$$p(s_i^m|\mathbf{x}) = \frac{p(s_i^m) \cdot p(\mathbf{x}|s_i^m)}{\sum_{n,j} p(s_j^n) \cdot p(\mathbf{x}|s_j^n)} \qquad (3.23)$$

Table 3.7: Training of pair-wise discriminant continuous HMM

	iteration of training		
	0	1st	2nd
PWD-C-HMM (prob. density)	79.9	82.7	82.8
	83.1	88.0	88.7
PWD-C-HMM (a-post. prob.)	80.3	83.1	83.1
	84.0	88.7	89.3
PWD-C-HMM (binary digit)	80.8	84.3	82.8
	84.4	89.0	88.2

upper row: open, lower row: closed

(3) binary digit

Here, 1 is assigned to the first candidate, and a small constant ϵ to others. This is a special case of a posteriori probability. Outputs of neural networks using sigmoid function often become this form practically.

$$\begin{cases} p(s_i^m|\mathbf{x}) = 1 & \text{if } s_i^m = s_{(1)} \\ p(s_i^m|\mathbf{x}) = \epsilon & \text{otherwise} \end{cases} \tag{3.24}$$

3.3.4 Experimental Evaluation

We have evaluated continuous HMM based on pair-wise discriminant analyses (PWD-C-HMM) on 27 phoneme (consonant) recognition. The number of the states is 3 or 4. The net number of the states or the classes including the vowel parts becomes 90. The number of the pair-wise classifiers was reduced to 970 by eliminating less confusing pairs.

We have evaluated the three forms of probabilities of PWD-C-HMM. Training was iterated three times, until the recognition score for testing samples converged. The recognition performance at each iteration is listed in Table 3.7. Actually, it converged at the first iteration as the classifiers were constructed initially using hand labeled data. The improvement at the first iteration comes from the correction of the hand labeling and the use of all segments of the speech samples.

It is compared with the conventional continuous HMM. Two types of HMM are tested. One uses single Gaussian distribution with a full covariance matrix, and the other uses 4-mixture distribution with diagonal covariances. Simple discriminant analysis is performed for them to reduce the dimension of input patterns. The phoneme recognition results are shown in Table 3.8. For open samples, PWD-C-HMM achieves about 13% higher recognition rate than the conventional continuous HMM gets. It should be noticed that

Table 3.8: Phoneme recognition result with continuous HMM

recognition method	closed	open
PWD-C-HMM (prob. density)	88.7	82.8
PWD-C-HMM (a posteriori prob.)	89.3	83.1
PWD-C-HMM (binary digit)	88.2	82.8
continuous HMM (single full covariance)	88.4	70.2
continuous HMM (4-mix. diag. covariance)	80.4	69.5

the recognition rates are much the same in closed experiments. It shows that pair-wise classifiers succeed in separating confusing classes as far as possible and realize a robust recognition system.

There is little difference between the three forms of probability normalization. This suggests that, in recognizing roughly segmented phonemes, ranking is more significant than scoring or values of output probabilities. However, the significance of stochastic scoring will be proved in word or sentence recognition.

From the computational viewpoint, PWD-C-HMM is not expensive although so many classifiers are set up. We can reduce the number of the pair-wise classifiers by eliminating less confusing pairs of the states. And in two-class discriminant analysis, the dimension of the vector gets smaller than that for many classes. Furthermore, we use linear Bayes classifier, which makes it possible to compute relative values of the probabilities without covariance matrices.

In our experiments, the average dimension of the vectors for pair-wise classifiers is 20, while the conventional HMM needs 62 variables. When we compare the amount of necessary computation for output probabilities, PWD-C-HMM needs 970 (pairs) $\times$ 2 (classes) $\times$ (20+1) (variables+constant) = 40740, while the conventional HMM with full covariances needs 90 (states) $\times$ 62 (variables) $\times$ 62 (variables) = 345960. Actually, our method takes less CPU time than the conventional HMM with full covariances.

3.3.5 Conclusion

We have proposed a new discriminative acoustic model named PWD-C-HMM that incorporates of simple and robust Bayes classifiers. It realizes discriminative recognition by fully separating competing pairs of the HMM states, and keeps stochastic scoring mechanism on the basis of statistical Bayesian decision. Experimental results show that our method is more effective than conventional continuous HMM.

3.4 Comparison of Discrete and Continuous Models

3.4.1 Introduction

We have studied the strategy to improve discriminating ability of both discrete and continuous HMM. It incorporates a classifier as front-end of HMM, instead of VQ or just passing directly that is the case for conventional continuous HMM. The classifier is trained to separate local patterns of the different phones, thus improves the separability of HMM. In this section, we make a comparison of its discrete distribution version and its continuous version, from the viewpoints oriented to classifier-based HMM. We also make experimental evaluation of the two models on various recognition tasks, using the same classifier but changing its characteristics.

3.4.2 Theoretical Comparison

In conventional HMM, the front-end processor just performs coding or feature extraction. In classifier-based HMM, large part of the recognition process is done by the classifier. Main role of HMM is dynamic time warping and post-processing of the classification. Therefore, comparison of discrete and continuous models must be done from the viewpoints different from those for conventional HMM.

We make a comparative review of the two versions in Table 3.9. The class or the target for the classifier in discrete model is the symbol that represents a local pattern of some phone, while the classifier in continuous model directly discriminates the HMM state that also represents a local pattern. The definition or the meaning of the output probability is different, too. In discrete model, the output probability shows the frequency of the symbol occurrence as in conventional HMM. However, the probability matrix is much more biased than that of VQ-based HMM, because each model has their belonging symbols explicitly and the probabilities of their occurrence become much higher. The symbol output probability matrix can be regarded as a confusion matrix of the classifier. Thus, even the tendency of the classification errors is learned by HMM. In continuous model, the output probability means the reliability of the classification. So the distribution of the probabilities depends on the characteristics of the classifier. It will be stochastic if the classifier is stochastic. In general, the classifier in discrete model is fundamentally independent from HMM, while the classifier in continuous model is tightly integrated with HMM.

Table 3.9: Comparison of discrete and continuous models

	discrete model (PWD-D-HMM)	continuous model (PWD-C-HMM)
class	symbol	HMM state
meaning of probability	frequency of symbol occurrence	reliability of classification
distribution of probability	biased	depends on classifier
training of classifier	independent from HMM	integrated with HMM

3.4.3 Experimental Comparison

Next, we made experimental evaluation of the two models. In order to make a fair comparison, we use the same classifier as well as the speech samples and the acoustic analysis method.

Training of Classifier and HMMs

Training of the classifier is iterated and interacted with that of HMM, so that the classifier and HMMs are fully integrated and totally optimized. Training algorithm is an iteration of the following steps. The flowchart is in Figure 3.8.

1. For every (continuous) HMM, state transition probabilities are estimated by Forward-Backward algorithm.
2. For all the training samples, Viterbi alignment is performed to assign each frame pattern vector to some state of the HMM.
3. For every pair of the states, feature extraction based on discriminant analysis is performed to construct a Bayes classifier, using the pattern vectors assigned in the previous step.

The parameters of continuous HMM are completely obtained by the above steps. For discrete HMM, we regard each HMM state as a symbol representing the local part of the phone. For example, we define phonetic element symbol *b1* for representing the first state of phone *b*. Then, the following steps are performed to get the HMM parameters.

1. For all the training samples, every frame is labeled by the classifier.

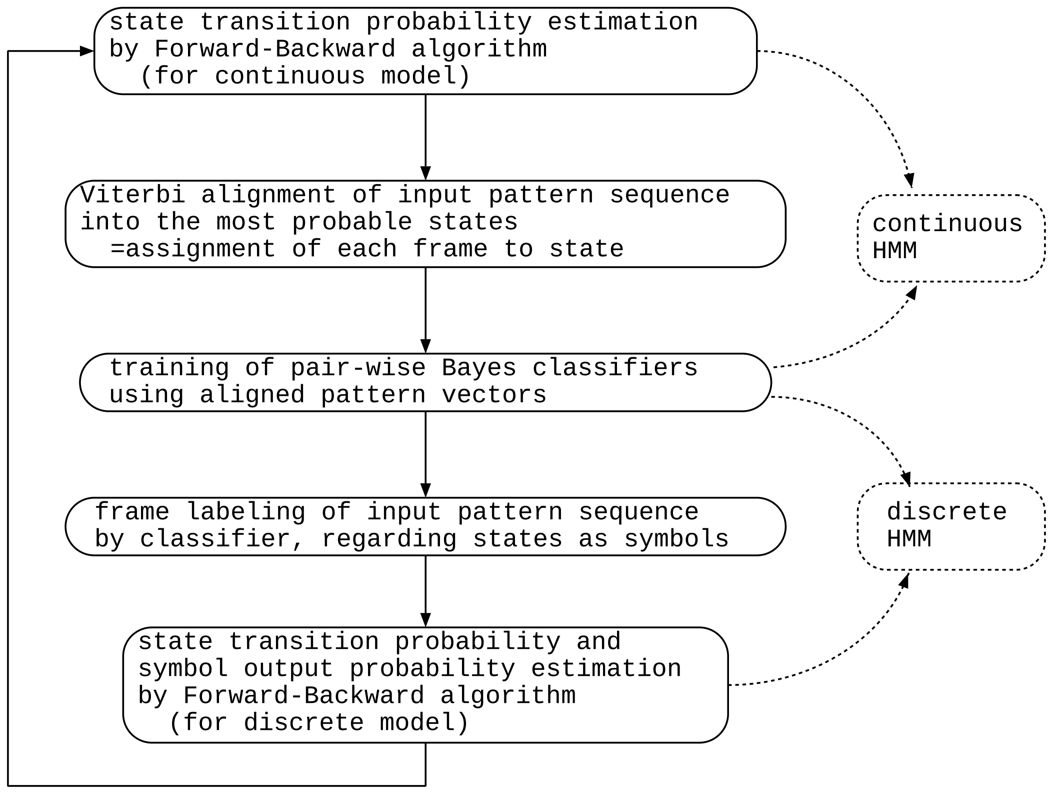
state transition probability estimation by Forward-Backward algorithm (for continuous model)
Viterbi alignment of input pattern sequence into the most probable states =assignment of each frame to state
training of pair-wise Bayes classifiers using aligned pattern vectors
frame labeling of input pattern sequence by classifier, regarding states as symbols
state transition probability and symbol output probability estimation by Forward-Backward algorithm (for discrete model)
continuous HMM
discrete HMM

Figure 3.8: Training of classifier and HMMs

2. For every (discrete) HMM, symbol output probabilities and state transition probabilities are estimated by Forward-Backward algorithm.

Experimental Condition

We set up 38 phone models: 28 for consonants and 10 for vowels and semi-vowels. The number of the states is 3 or 4 for consonants and 1 for vowels. The net number of the states or the phonetic element symbols becomes 97. The number of the pair-wise classifiers was reduced to 1116 by eliminating less confusing pairs.

We have performed recognition of phonemes, words and sentences. Since HMMs are constructed just for phonemes, word or sentence recognition is done by concatenating the phone HMMs according to the lexical and the syntactic transcription. For phoneme recognition and word recognition, we use the same data set, which is divided into training samples and testing ones. Those experiments are done in multiple speaker mode, but the speakers for sentence recognition are independent.

Table 3.10: Phoneme recognition result for acoustic model evaluation

	iteration of training		
	0	1st	2nd
PWD-D-HMM (normal)	82.4 93.1	85.1 95.2	86.3 94.5
PWD-D-HMM (fuzzy)	83.6 89.5	85.5 91.1	86.3 91.7
PWD-C-HMM (prob. density)	79.9 83.1	82.7 88.0	82.8 88.7
PWD-C-HMM (a-post. prob.)	80.3 84.0	83.1 88.7	83.1 89.3
PWD-C-HMM (binary digit)	80.8 84.4	84.3 89.0	82.8 88.2
VQ-based discrete HMM (single codebook)	61.7 81.1		
VQ-based discrete HMM (multiple codebook)	67.7 89.4		
normal continuous HMM (single full covariance)	70.2 88.4		
normal continuous HMM (4-mix. diag. covariance)	69.5 80.4		

upper row: open, lower row: closed

Phoneme Recognition

Phoneme recognition task here is to recognize the consonant of 27 kinds contained in the VCV pattern. We have applied VQ-based discrete HMM and conventional continuous HMM as well as classifier-based discrete and continuous HMM. As for VQ-based HMM, we have tested two types. One uses single codebook of size 256, and the other uses separate codebooks for spectrum and power[11], with 256 and 64 entries respectively. We have also tested two types of conventional continuous HMM. One uses single Gaussian distribution with a full covariance matrix, and the other uses 4-mixture distribution with diagonal covariances.

The recognition results are listed in Table 3.10. Notation ‘PWD-D-HMM’ stands for pair-wise discriminant discrete model and ‘PWD-C-HMM’ for continuous one. As for classifier-based HMM, recognition rates at each training iteration are presented. They converged after the first iteration because the classifier was available initially. For open samples, classifier-based HMM achieves much higher recognition rates than conventional

Table 3.11: Word recognition result for acoustic model evaluation

	iteration of training		
	0	1st	2nd
PWD-D-HMM (normal)	80.4	81.0	80.4
PWD-D-HMM (fuzzy)	84.4	83.8	85.0
PWD-C-HMM (prob. density)	78.0	83.6	84.7
PWD-C-HMM (a-post. prob.)	79.3	84.0	84.8
PWD-C-HMM (binary digit)	72.7	79.2	78.9
normal continuous HMM (single full covariance)	72.5		

discrete and continuous HMM get. It should be noticed that the recognition rates are much the same in closed experiments. It shows that the pair-wise classifiers succeed in separating confusing classes as far as possible and realize a robust recognition system, while conventional HMM does not explicitly try to separate different classes.

Among classifier-based models, discrete version generally gets higher rates than continuous version regardless of the output forms. In recognizing roughly segmented phonemes, ranking of the classes is more significant than scoring or values of the output probability. Therefore, discrete version that learns symbol patterns including the tendency of the classification errors gets superior.

Word Recognition

Word recognition experiments are performed on 653 vocabulary size. The average number of the phonemes in a word is 4.69.

The results are listed in Table 3.11. The recognition rates converged after the first iteration of training of the phone HMMs. Classifier-based HMM is more effective than conventional continuous HMM in any cases.

This time, however, there is much difference among the output types of the classifier. In discrete version, fuzzy labeling is more effective than normal deterministic labeling. In continuous version, using binary digit is harmful. These suggest that stochastic outputting or real-value scoring is significant. In word recognition, the number and the locations of the phonemes are unknown, thus digital outputs make it difficult to deal with the time alignment problem. In this case, discrete model with fuzzy labeling is comparable to continuous model with normal probabilities.

Table 3.12: Sentence recognition result for acoustic model evaluation

	rec.	com.	word
PWD-D-HMM (normal)	37.3	55.0	77.2
	67.0	79.0	87.8
PWD-D-HMM (fuzzy)	46.5	64.8	84.5
	75.5	83.5	93.0
PWD-C-HMM (prob. density)	44.0	57.3	79.7
	71.0	79.3	89.2
PWD-C-HMM (a-post. prob.)	51.3	63.7	84.0
	75.5	82.0	92.8
PWD-C-HMM (binary digit)	41.5	56.0	76.2
	66.5	74.3	85.0

upper row: top-1, lower row: top-10

Sentence Recognition

For sentence recognition, we set up a task named "personal schedule management at workstation". The word perplexity is 37.7 and the average number of the words in a sample sentence is 6.2. Recognition is performed by A* search that is controlled by a context free grammar described for the task[52].

The results are listed in Table 3.12. Here we use three evaluation measures. The sentence recognition rate (rec.) is a ratio of the samples whose word sequences are completely matched. In the sentence comprehension rate (com.), we count as correct if all the keywords that constitute the meaning for the task are completely recognized. We also investigate the recognition rate of such keywords in the sentences (word). 10-best algorithm is performed to make careful evaluation.

We find much the same tendency as that in word recognition. The methods which perform stochastic outputting are more effective than those which do not. In most evaluation measures, continuous model with a posteriori probability is comparable to discrete one with fuzzy labeling, and continuous model with binary digit is comparable to normal discrete one. By the top-1 sentence recognition rate, however, we see a clear ranking of the five methods: a posteriori probability, fuzzy discrete, probability density, binary digit, normal discrete.

Table 3.13: Summary of recognition results for acoustic model evaluation

task	phoneme	word	sentence
PWD-D-HMM (normal)	86.3	80.4	37.3
PWD-D-HMM (fuzzy)	86.3	85.0	46.5
PWD-C-HMM (prob. density)	82.8	84.7	44.0
PWD-C-HMM (a-post. prob.)	83.1	84.8	51.3
PWD-C-HMM (binary digit)	82.8	78.9	41.5

open experiments at final iteration of training

Discussion

Table 3.13 shows the summary of the experiments. It is noticeable that the tendency of phoneme recognition is different from those of word and sentence recognition. To improve discriminating ability on phoneme recognition does not necessarily lead to a high performance on word or sentence recognition.

In continuous speech recognition, stochastic outputting is so important as, or even more important than, to improve the discriminating ability too much. Even in continuous model, it is harmful to use probabilities that are substantially binary. And in discrete model, we can realize a high recognition rate simply by passing multiple candidates.

3.4.4 Conclusion

We have compared discrete and continuous versions of classifier-based HMM. They have different meanings of the output probability since the target of the classifier is different. However, experimental results show that the difference of the two models is not so significant as the difference of the classifier's output characteristics. It is confirmed that, in continuous speech recognition, really stochastic distribution of the outputs is the most important.

Chapter 4

Continuous Speech Parsing based on Heuristic Search

Continuous speech recognition is regarded as searching for the most plausible phone or word sequence that satisfies the linguistic and the pragmatic constraints.

As the constraint, several knowledge sources such as lexicon, syntax, semantics and pragmatics are available. It is natural that use of more constraints will bring better accuracy. It is desirable that multiple-level knowledge sources are jointly utilized as an integrated processing, which is illustrated in Figure 4.1.

Although simple uniform integration of multiple-level knowledge sources[53] is idealistic, it causes computational explosion, since the number of possible sentences is enormous, and so is the possible combination of the constraints. Thus, the modularity of the knowledge sources is also significant for practical computation and easy maintenance.

We adopt a heuristic search strategy, which maintains the modularity of the knowledge

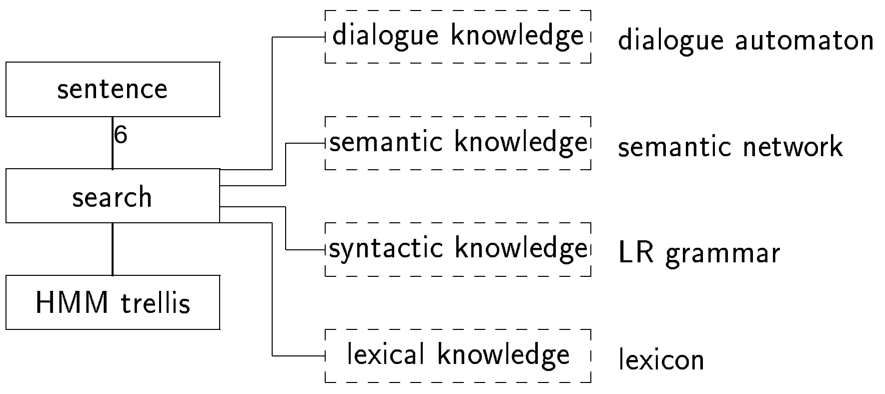

Figure 4.1: Framework of search integrating multiple knowledge sources

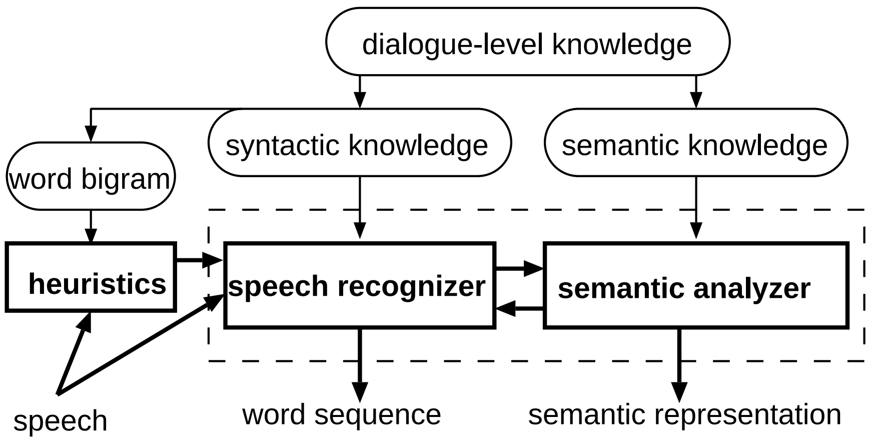

Figure 4.2: Heuristic search strategy integrating multiple knowledge sources

sources and guarantees the admissibility to find the optimal hypothesis. Its outline is shown in Figure 4.2. It first performs rough decoding with a simple constraint such as word bigram, and then applies higher-level linguistic constraints using the result of the first pass as A*-admissible heuristics. We basically use syntactic knowledge to generate hypotheses at the speech recognizer and semantic knowledge to verify them at the semantic analyzer. The both knowledge sources are constrained by dialogue-level knowledge.

The search strategy makes it possible to obtain the optimal hypothesis that satisfies all the constraints. The key is to choose a good constraint which provides admissible and powerful heuristics.

In Section 4.1, we present search algorithms with syntax only. In Section 4.2, dialogue-level knowledge is incorporated. In Section 4.3, use of semantic knowledge is discussed.

4.1 A* Algorithm for LR Parsing with HMM Front-End

4.1.1 Introduction

In spoken language understanding, an efficient search algorithm is indispensable, since the number of possible sentences is enormous, given a restricted grammar. In the search algorithm, it is significant both not to lose the optimal hypothesis and to extend least hypotheses. Beam search, which is conventionally used, makes evaluation based on the score to the searched point, and may prune the optimal hypothesis due to the local mismatch. It must deal with as many hypotheses as the beam width that is proportional to the task perplexity.

In this section, we discuss heuristic search algorithms that incorporates estimate score of the unsearched parts, especially on A* algorithm that is guaranteed to get the optimal hypothesis. The key to realize A* search is an effective heuristic function. Therefore, we examine requirements on the heuristics, then propose lexical constraint, especially word-pair constraint as heuristics. We also discuss incorporating heuristics into beam search and examine the effects of heuristic power on the search methods, from the viewpoints of both the accuracy and the efficiency.

As the constraint for the search, here, we use a lexicon and an LR (context-free) grammar. LR parsing is necessary for the conversational utterances. We also extend it to the stochastic version.

4.1.2 A* Search in Continuous Speech Recognition

Continuous Speech Recognition as Search Problem

In order to make direct use of linguistic constraints, the search space should be the pattern space which the phone model deals with. When we use HMM as a phone model, it is an HMM trellis. This time-state space, as shown in Figure 4.3, is an expansion of HMM states to the time-dimension according to the permissible state transitions. Each state at each time has an output probability that the input speech belongs to the phonetic element state at that time, and a state transition probability is assigned to each connection.

For a finite state automaton grammar, it is possible to prepare a dedicated trellis where all the states are expanded. For a context-free grammar, however, a trellis is generated for each hypothesis by concatenating phone HMMs according to the lexical and the syntactic transcription.

The evaluation function for a sentence hypothesis is computed by Viterbi algorithm that realizes score maximization over the trellis paths representing the sentence. The score is a summation of the output probabilities and the transition probabilities on logarithm scale. It is a negative value. In addition to the bottom-up score, it is possible to incorporate a top-down score computed by a stochastic language model.

Thus, we formulate the search problem that finds the most plausible sentence hypothesis on the trellis space under given linguistic constraints.

As for the direction of the search, left-to-right search is considered. In this case,

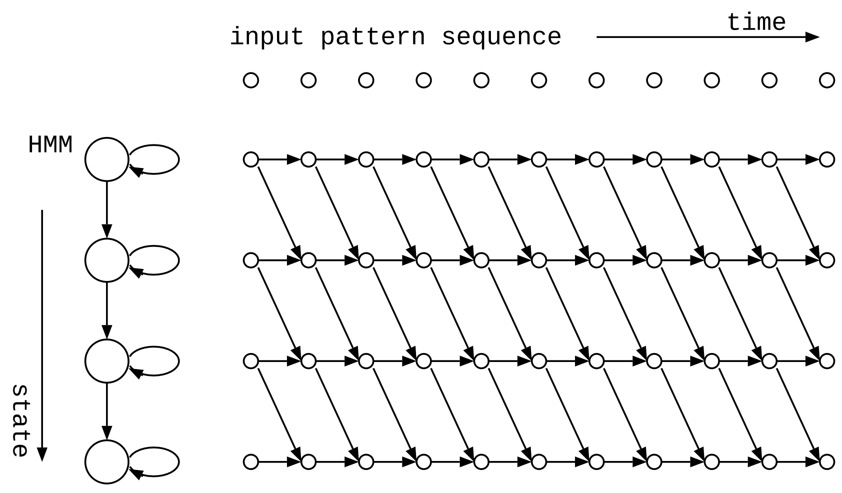

Figure 4.3: Trellis generated by HMM

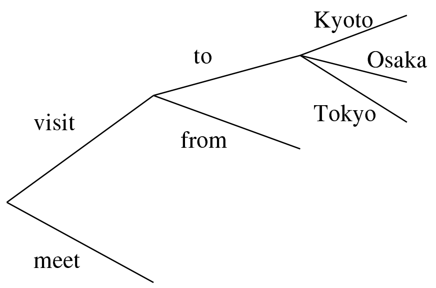

Figure 4.4: Word tree for sentence hypotheses

sentence hypotheses are represented as paths on the word tree that is expanded to the time-dimension, as shown in Figure 4.4. Actually, we perform a tree search, using Viterbi computation on HMM trellises that correspond to the tree paths.

Heuristic Search

A* search is a kind of best-first search. It proceeds by extending the hypothesis whose current evaluation score is the largest. It is significant to incorporate estimate of the unsearched part. Thus, the evaluation function for a sentence hypothesis n is defined as the sum of the matching score of the extended part $g(n)$ and the heuristics $\hat{h}(n)$. The search procedure is illustrated in Figure 4.5.

$$\hat{f}(n) = g(n) + \hat{h}(n)$$

In order to guarantee the search A*-admissible, or to get the optimal hypothesis, heuristics $\hat{h}(n)$ must be an estimate of the best scoring extension to the end of the input speech.

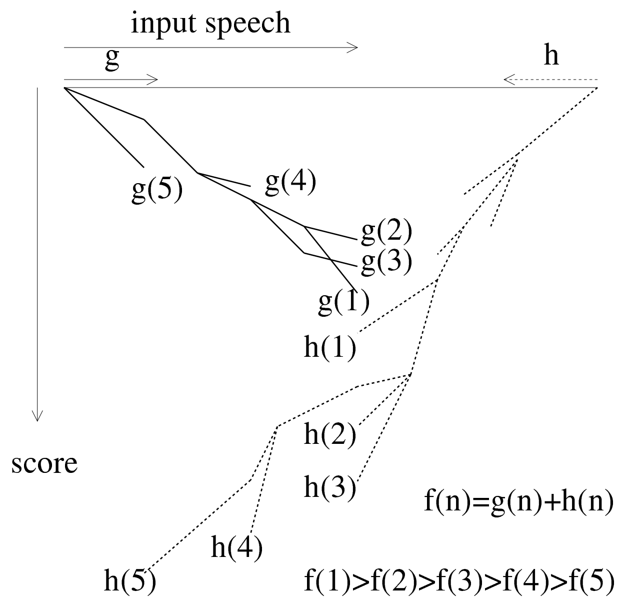

Figure 4.5: Scoring of A* search

Namely, the estimate has to be more optimistic than actual[54].

Now we compare A* search with beam search. Beam search is a complex of breadth-first search and best-first search. It extends all the hypotheses step by step, while putting away those with lower scores in order to limit the number of the nodes to a certain beam width. Frame-synchronous search proceeds along with the time-frame of the inputs, while frame-asynchronous search directly traverses a word tree or a phone tree.

Therefore, the globally optimal hypothesis can be abandoned if its score gets lower locally. In order to avoid such cases, the beam width should be large enough according to the task perplexity. But enlarging the beam width implies that the search must deal with so many hypotheses from the beginning to the end.

A* search extends less hypotheses if it is guided by an adequate heuristics. However, if heuristics is so weak, the search gets breadth-first and fails to complete any hypotheses. To manage the search within realistic computation, stack decoding[28] is used to limit the number of hypotheses. Combined with a weak heuristics, however, it may lose the optimal hypothesis. When heuristics is extremely weak, it results in much the same as beam search. In short, realization of A* search depends on its heuristics. This issue is discussed later.

It is also possible to incorporate heuristics into beam search. It uses the same evaluation function, but proceeds breadth-first within a beam width. Heuristics will guide the globally optimal hypothesis and protects it from pruning. Thus, the search will succeed with a smaller beam width. Namely, adequate heuristics makes beam search effective, too.

Related Works

A* search for continuous speech recognition was studied by W.A.Woods[55] and others about a decade ago. Their strategies assumed that the search space is a word lattice, where the appearances and the positions of words are given, and the number of possible sentence hypotheses is small enough. Their heuristics computation also made use of word recognition results. Therefore, they are not applicable to parsing on HMM trellis, which generates far larger number of sentence hypotheses.

Recently several studies on A* search on HMM trellis is appearing. D.B.Paul's stack decoder[56] first assumed no language modeling on heuristic function, thus very weak for searching. Incorporating a simple stochastic language model does not satisfy admissibility[57]. F.K.Soong's strategy[58] assumes a finite state grammar, and is effective for finding N-best hypotheses, not the first candidate itself. However, context-free parsing is necessary at the task which is not so simple, and it is highly troublesome to describe an automaton grammar.

The similar forward-backward search algorithm is proposed by S.Austin[59]. It is fundamentally beam search that uses the result of the first pass for pruning and heuristic scoring. It is not admissible.

Here, we focus on A* algorithm for LR parsing that directly drives phone models to compute evaluation function, and deals with all the possible hypotheses without any intermediate representations.

4.1.3 Word-Pair Constraint as Heuristics

Requirements for Heuristics

Here we examine the requirements for the heuristic function $\hat{h}(n)$ to realize successful A* search.

- Admissibility

In order to guarantee the search to find the optimal solution, the estimate must be more optimistic than the actual score, namely $|\hat{h}(n)| \geq |h(n)|$.

- Search efficiency

  In order to reach the optimal solution with least node expansions, the estimate should be as accurate as possible, namely $\hat{h}(n) \simeq h(n)$.

- Heuristics computation

  The estimation should be computationally cheap, because the overall efficiency is evaluated by the search efficiency itself plus the heuristics computation.

Useful Constraints for Heuristics

Considering the above requirements, then we examine which constraints are appropriate for the heuristics computation. When using a context-free grammar, every sentence hypothesis must be tested if it satisfies the constraint one by one using dedicated stack. But the verification is impractical for the heuristics estimation. A finite state grammar makes it possible to process all the hypotheses with a single automaton model. However, a large-scale grammar needs much computation, too. Lexical constraint is common to all the hypotheses.

Therefore, we use, as heuristics, word-pair constraint that reflects lexical constraint and simple syntactic constraint. It is represented by a parallel concatenation of word models, where syntactically admissible pairs of the words are connected. We call this word-pair HMM. It is exemplified in Figure 4.6. The regular grammar that decides which words can follow which words is derived so that it accepts a superset of the language generated by the original context-free grammar. This guarantees the A*-admissibility condition. Moreover, the word-pair constraint, which is a first-order approximation of the grammar, is expected to be powerful heuristics for the search.

Search (Parsing) Algorithm

In order to compute the heuristic scores for all the possible partial sentence hypotheses, overall procedure consists of heuristic computation and the search itself. These two passes are performed in different directions. When we take left-to-right search, we compute

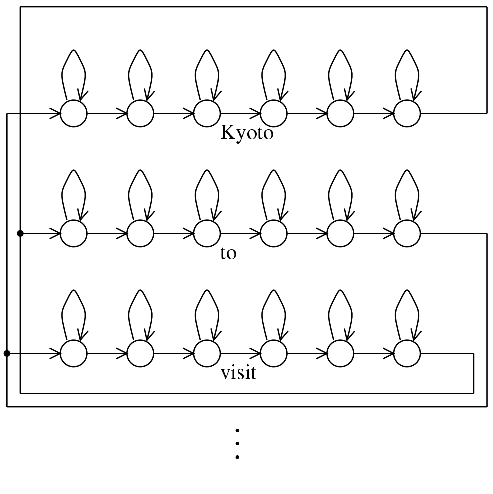

Figure 4.6: Word-pair model

heuristics right-to-left beforehand. Actually, the reverse direction is more practical as we can compute heuristics left-to-right in real time.

The word-pair model is applied from the end of the input speech to the beginning, and a backward trellis is generated. The heuristic score from any word node $(S_w, t)\{S_w\text{:state}, t\text{:time}\}$ to the end of the speech is computed and stored. The scores are commonly used for all the partial sentence hypotheses.

In searching, a forward trellis is generated for each sentence hypothesis by concatenating HMMs according to the transcription. Suppose the last word of partial sentence hypothesis n and its first HMM state are a and S_a^n respectively, and the first HMM state of word a in the word-pair model is S_a^w, then the evaluation function $\hat{f}(n)$ is defined by the recognition score $\alpha_n(S_a^n, t)$ computed on the forward trellis and the heuristic score $\beta_w(S_a^w, t)$ on the backward trellis.

$$\hat{f}(n) = \max_{1 \le t \le T} (\alpha_n(S_a^n, t) + \beta_w(S_a^w, t))$$

$$T : \text{number of time points in whole trellis}$$

Here $\alpha_n(S_a^n, t)$ and $\beta_w(S_a^w, t)$, computed by Viterbi algorithm, represent $g(n)$ and $\hat{h}(n)$ respectively. The above evaluation performs concatenation of the forward trellis and the backward trellis, which is shown in Figure 4.7. In the example of the left upper-most in Figure 4.7, a hypothesis "visit to" is generated, where the trellis of the last word "to" and the following heuristics come from the common backward trellis shown right-hand. This evaluation is computationally cheap, and the actual trellis computation for the last word a is performed only when the hypothesis is popped from the stack.

In the forward search, the whole linguistic constraints are used to limit the search space, namely to eliminate vain hypotheses at an early stage. In this section, LR grammar is used to predict the words that can extend the present partial sentence, and to check if the sentence is completed and accepted at the end. Since heuristics is admissible, the first hypothesis that is accepted by the grammar and reaches the end of the input speech (on that case, T gives the maximum in the above formula) is guaranteed to be the optimal solution.

The search algorithm is described below.

1. A backward trellis is generated to compute the heuristic scores with the word-pair model.
2. Words that can appear at the beginning of sentences are picked up. For each of them, a new sentence hypothesis is generated and pushed to the stack with its evaluation score.
3. The hypothesis n with the largest score is popped from the stack.
4. If the hypothesis n reaches the end of the input speech and is accepted by the given LR grammar, then it is the optimal solution. Finish the search.
5. The trellis for the last word of the hypothesis n is extended.
6. Words that can extend the hypothesis n are predicted. For each of them, a new sentence hypothesis replacing n is generated and pushed to the stack with its evaluation score. Go to Step 3.

This is easily extended to an N-best algorithm. If we continue the search after finding the optimal solution, we can obtain the N-best candidates correctly.

Extension to Stochastic LR Parsing

In the above algorithm, the syntactic constraint is used only to limit the search space. To realize more plausible understanding, we introduce a stochastic LR grammar. The combination of the linguistic statistics and the acoustic score is formulated by Bayesian rule.

Each rewriting rule r of the grammar has a priori probability $p(r)$ that indicates how often it is applied. The probabilities are normalized so that the sum of those having an

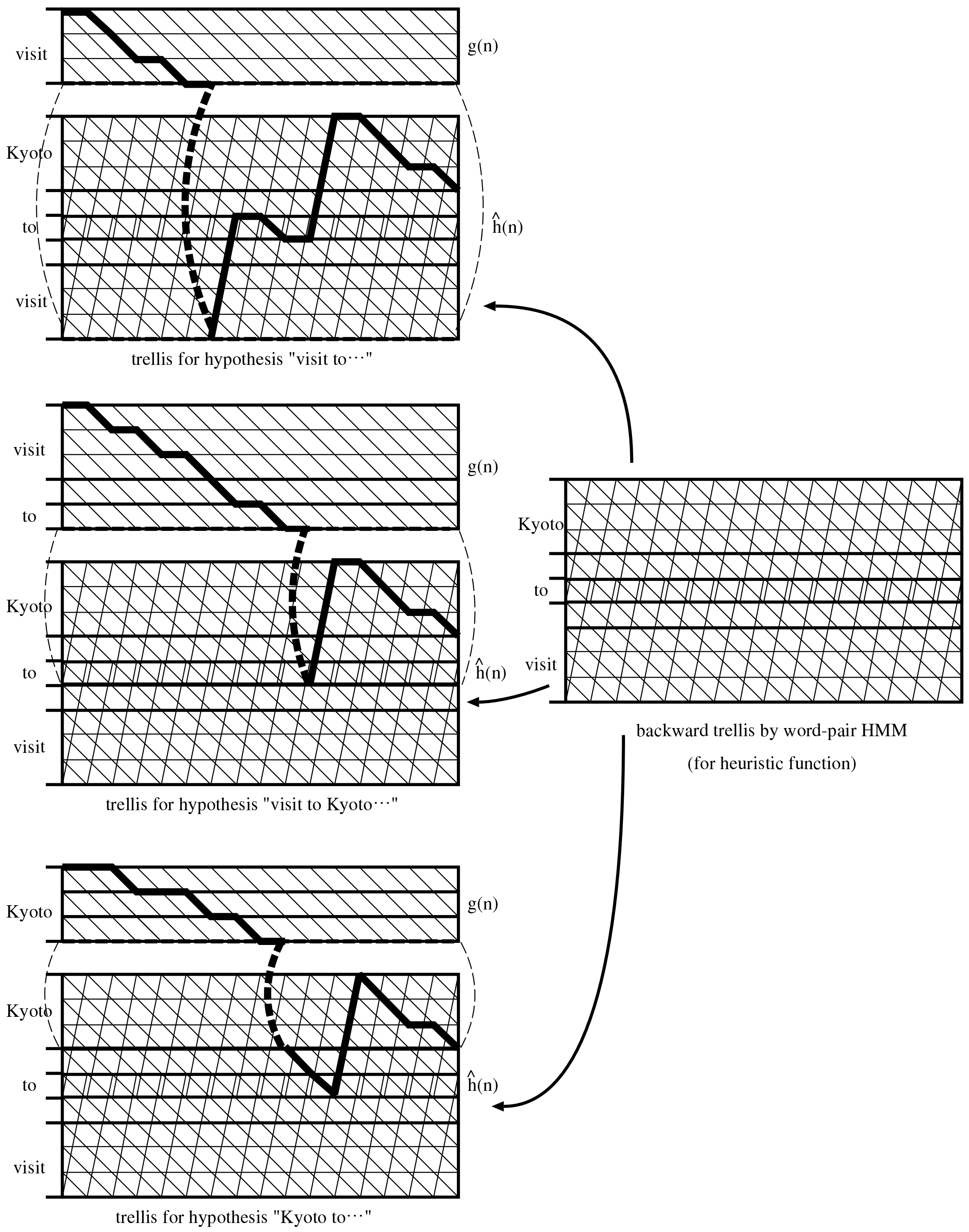
visit
g(n)
Kyoto
to
visit
ĥ(n)
trellis for hypothesis "visit to···"
visit
to
g(n)
Kyoto
to
visit
ĥ(n)
trellis for hypothesis "visit to Kyoto···"
Kyoto
to
visit
backward trellis by word-pair HMM
(for heuristic function)
Kyoto
g(n)
Kyoto
to
visit
ĥ(n)
trellis for hypothesis "Kyoto to···"

Figure 4.7: Virtual concatenation of forward trellis and backward trellis

identical left non-terminal symbol becomes 1. In recognition, every time some rule is applied, its a priori probability is added to the HMM score on logarithm scale. Thus, a priori probability for word sequence W is

$$\log p(W) = \sum_i \log p(r_i(W)) \tag{4.1}$$

where $r_i(W)$ is the i-th rule that is applied for W.

To apply this grammar to A* search, the heuristics computation is modified so that its constraint is subset of, but close to the original LR grammar. We incorporate the word transition probability to the word-pair grammar. The conventional word bigram that is computed independently of linguistic knowledge may be powerful, but does not satisfy admissibility. Therefore, we derive the probability from the original stochastic grammar.

The word transition probability from word a to word b is defined by a priori probability of the rewriting rule that is reduced when b is input after a. Since the stack state after reading word a is not unique in LR grammar, we take the maximum of the probabilities for all the possible reduction paths.

This transition probability provides an estimate of $\log p(W)$. Thus, we obtain word bigram, to be exact probabilistic word-pair, as A*-admissible heuristics.

4.1.4 Experimental Evaluation

The parsing algorithm is evaluated on speaker-independent continuous speech recognition. The task domain is "personal schedule management at workstation". The sentence utterances are concerning registration, modification and query on the schedule database. They are uttered continuously without explicit pauses between words. We picked up 50 kinds of sample sentences, each of which was uttered by 8 male speakers who are not among those for training the phone models. Phone models are based on discrete HMM with pair-wise symbol classifiers.

Here, we used 2 kinds of LR grammars, which are different in terms of perplexity and coverage. Grammar G0 allows a user more flexible utterances than grammar G1. For each of them, word-pair grammar is derived as heuristic constraint. The word perplexity of each constraint is listed in Table 4.1. It shows that the word-pair grammar approximates well the original LR grammar and will work as a powerful heuristic constraint. Especially for phrase recursive grammar G0, the perplexity of word-pair is very close to that of the LR grammar.

Table 4.1: Word perplexity with syntactic and heuristic constraint

	G1	G0
LR grammar	17.2	37.7
word-pair	32.7	39.5
no grammar (word)	223.0	208.0

In the experiment, we limited the stack size to 160 and the total number of the hypothesis extension to 300 to leave off the hopeless search. However, there was no case in the experiment where a hypothesis once at the bottom of the stack is popped down. The search is continued until the 10 best candidates are found. For comparison, we also performed beam search, both frame-synchronous and frame-asynchronous. Frame-asynchronous search is word-synchronous, namely proceeds word by word using the evaluation score normalized by the frame length. They are performed with several variety of beam widths (stack sizes).

The recognition results are listed in Table 4.2. Here we use three evaluation measures. The sentence recognition rate is a ratio of the samples whose word sequences are completely matched. In the sentence comprehension rate, we count as correct if all the keywords which constitute the meaning for the task are completely recognized. We also investigated the recognition rate of such keywords in the sentences.

In any cases, A* search achieved higher performance than beam search did. Beam search got better accuracy as the beam width is larger. But it lost 5∼10% of optimal hypotheses in the best cases. Though strong grammar G1 brought the better accuracy than grammar G0, the same tendency is confirmed with respect to the search methods. The recognition or comprehension errors were more likely to occur in rather longer sentences, quite naturally. A large part of the recognition errors were concerning digit words as they are free from the contexts and acoustically confusing.

In order to make computational evaluation of the search algorithms, we investigated the number of generated nodes (gene.), pushed nodes (push), and poped nodes (pop). Averages per an input sample are listed in Table 4.2. For each algorithm, the relation between these numbers and computational amount is different. Since the most expensive process is HMM trellis computation, we compare the number of nodes whose corresponding trellises are extended.

In A* search or heuristic search, heuristics computation needs extra trellis computa-

Table 4.2: Comparison of search algorithms in continuous speech parsing

search method	rec. (%)	com. (%)	word (%)	gene. node	push node	pop node
A*	63.0	68.0	81.4	4256	927	110
(160)	42.5	56.5	80.5	5160	851	60
asynchro-beam	55.0	61.8	81.9	6003	635	302
(40-30)	33.3	47.5	74.6	20492	801	327
asynchro-beam	58.3	65.5	84.1	10534	1355	617
(80-60)	37.8	51.5	77.4	39092	1830	676
synchro-beam	52.5	57.3	77.2	8138	7528	263
(160-60)	33.8	46.5	75.4	11974	11391	175
synchro-beam	59.8	65.0	84.1	17639	16403	499
(320-120)	37.8	52.5	78.7	28623	27561	298
synchro-beam	60.3	65.8	84.7	27105	25150	722
(480-180)	39.3	54.0	79.7	46856	45291	410

upper row: G1, lower row: G0

search method : (global stack size - local beam width)

tion. Approximately, it is defined by the vocabulary size and the number of words in an input. Their product in this case gets 1400 (=225×6.27). This amount has to be counted. In the forward search, however, the trellis for the very last word of a hypothesis is given by the heuristics and is not generated until poped from the stack again.

On the other hand, the conventional blind search without heuristics extends the trellises as it generates hypotheses. Therefore, in frame-asynchronous beam search, the overall computation is proportional to the number of generated hypotheses. In frame-synchronous search, completely extended hypotheses are poped ones, while partially extended hypotheses may be pruned. Thus, the computation is proportional to a midst of the pushed and poped hypotheses.

Even when considering heuristics computation, A* search needed least trellis generation. Actually, the CPU time of the A* search is less than half of the frame-synchronous beam search. Beam search deals with as many hypotheses as the beam width, thus the number of hypotheses and CPU time is proportional to the beam width. Frame-asynchronous search needs smaller beam width, but more trellis computation. From the computational point, phone-synchronous search would be better.

Next, we investigate the effects of heuristics itself. We incorporated heuristics into beam search. We also performed best-first search without heuristics. In this case, the

Table 4.3: Effect of heuristics in continuous speech parsing

search method	constraint (g+$\hat{h}$)	$\lvert G\rvert$	$\lvert H\rvert$	rec. (%)	com. (%)	word (%)	pop node
A* (160)	G1 + word-pair	17.2	32.7	63.5	68.5	81.9	110
A* (160)	G0 + word-pair	37.7	39.5	42.5	56.5	80.5	60
beam (10)	G1 + word-pair	17.2	32.7	62.8	67.8	82.2	79
beam (10)	G0 + word-pair	37.7	39.5	42.8	57.0	81.1	47
beam (20)	G1 + word-pair	17.2	32.7	63.0	68.0	84.4	167
beam (20)	G0 + word-pair	37.7	39.5	42.5	56.5	81.0	171
A* (160)	G1 + word	17.2	223.0	62.5	67.0	77.0	254
A* (160)	G0 + word	37.7	208.0	42.3	55.5	74.3	178
beam (20)	G1 + word	17.2	223.0	61.8	66.0	79.7	181
beam (20)	G0 + word	37.7	208.0	41.8	55.8	80.2	182
beam (30)	G1 + word	17.2	223.0	61.8	66.8	81.7	271
beam (30)	G0 + word	37.7	208.0	42.0	56.0	80.5	277
beam (80)	G1 + (none)	17.2	-	58.3	65.5	84.1	617
beam (80)	G0 + (none)	37.7	-	37.8	51.5	77.4	676
best-first (160)	G1 + (none)	17.2	-	45.8	58.0	74.8	155
best-first (160)	G0 + (none)	37.7	-	29.0	44.8	63.9	133

$\lvert G\rvert$: perplexity with gramamr, $\lvert H\rvert$: perplexity with heuristics $\hat{h}$

evaluation score is defined by the forward score normalized by the frame length. In heuristic search, whether best-first or beam, the computation depends on the number of poped hypotheses. In blind search without heuristics, it depends on the number of generated hypotheses. Furthermore, we also examined the heuristic power. Besides word-pair constraint, we tested simple word-concatenation constraint where any word connections are allowed.

The recognition results are listed in Table 4.3. Since the constraint can be measured by the word perplexity, the values are shown for both original grammar $\lvert g\rvert$ and heuristic constraint $\lvert\hat{h}\rvert$. The heuristic power can be defined by the ratio or difference of the two values.

In beam search, too, incorporating heuristics protects optimal hypotheses from pruning with much smaller beam width. On the other hand, best-first search without heuristics is the worst as it cannot recover the errors at the beginning part of the speech. These results show heuristics improves both the accuracy and the efficiency, regardless of the

Table 4.4: Effect of stochastic grammar in A* search

	rec. (%)	com. (%)	word (%)
normal grammar	65.0	70.5	86.2
	42.5	56.5	80.5
probabilistic grammar	67.0	73.5	88.2
	46.5	61.0	82.6

upper row: G1, lower row: G0

search methods.

A* search with weak heuristics such as word-concatenation sometimes failed to complete any hypotheses, generating too many hypotheses. Such cases are more likely to occur in noisy samples that are not correctly recognized in any ways. Therefore, the sentence recognition rate is much the same as the best case, but the word recognition rate gets worse. Beam search avoids this problem by suppressing the number of generated hypotheses to a certain width.

With powerful heuristics such as word-pair, the search almost always obtained the optimal hypothesis. In this case, A* search is superior as it generates less vain hypotheses.

When we compare the two grammars G0 and G1, more constrained grammar G1 have smaller search space and needed smaller hypothesis expansions in the conventional search without heuristics. However, in heuristic search, where the heuristic constraint is much the same for the both grammar, the heuristic power gets relatively weak for grammar G1. Thus, more hypotheses are generated with grammar G1.

In the sentence parsing, it is assumed that weaker heuristics than word constraint is hopeless especially for A* search, as the search sometimes fails with this heuristics.

Finally, we implemented parsing with a stochastic LR grammar. Ideally the probabilities should be estimated using a large corpus, but there were no appropriate corpora available. Therefore, we assigned completely a priori value, namely from our insight. The results are listed in Table 4.4. Using stochastic LR grammar, even with completely a priori values, improved recognition accuracy.

4.1.5 Conclusion

We have presented an A* search algorithm for context-free parsing of continuous sentence speech. This best-first search evaluates each sentence hypothesis with the matching score plus the heuristic score of the unsearched part. Heuristics computation is based on lexical constraint that is common to all the hypotheses. Word-pair constraint, which is not computationally expensive, satisfies A*-admissibility and provides powerful heuristics. The algorithm is guaranteed to get the optimal hypothesis and does not expand vain hypotheses so much. Compared with various kinds of beam search without heuristics, therefore, it achieved better accuracy with less computation.

We have also examined effects of heuristic power. Heuristics turned out effective for beam search, too. With weaker heuristics, A* search sometimes fails and beam search is safer. With strong heuristics such as word-pair, A* search is superior with respect to the efficiency.

In this section, we explained our algorithm in left-to-right search. However, it is quite straightforward to implement it in right-to-left search. Right-to-left search is advantageous in its speed, as it can compute heuristics left-to-right almost in real time.

4.2 Using Dialogue-level Knowledge for Speech Parsing

4.2.1 Introduction

In spoken dialogue systems, it is significant to make use of dialogue-level knowledge for understanding user utterances. A goal-oriented dialogue between a user and the system, as in spoken dialogue system, proceeds in several typical patterns, which are modeled with scripts or plans and so on. Therefore, it is possible to constrain what the user utters next from the previous sequence of the utterances.

Moreover, it is desirable to apply dialogue-level constraint in speech recognition phase, not to the result of the speech recognition. Namely, it should be used to predict next user utterances and reduce its perplexity, not to post-process the recognized candidates. In this paper, we focus on predicting the syntactic type of a next user utterance, as it is more directly related with grammar-based speech parsing phase.

We propose a method to utilize the dialogue-level constraint for reducing syntactic perplexity of the user utterances and show its experimental evaluation.

4.2.2 Syntactic Prediction with Dialogue-level Knowledge

There have been several works to use dialogue-level knowledge sources to improve speech understanding[60],[61]. However, most approaches apply the dialogue-level constraint to sentence or phrase level that is the results of the speech recognition units. There is limitation in such strategy as it just chooses a plausible hypothesis among given candidates and cannot recover if the correct one is not in candidates. It does not contribute to the speech recognition by limiting the search space and eliminating the invalid hypotheses.

Hauptman et al. proposed to use script-based dialogue-level knowledge sources to predict concepts of next user utterances which are dynamically transformed to a semantic grammar network (RTN)[62]. He demonstrated to improve speech recognition accuracy with dialogue-level knowledge. However, the mechanism is too elaborate to implement by human or automatically, namely to derive complex structures and describe scripts and so on. It is desirable to build prediction mechanism easily and even automatically, given a large corpus. Furthermore, the predicted network grammar is completely task-dependent and allows only fixed patterns of sentences. At least a grammar of context-free class is necessary to describe conversational speech.

Here we propose to predict grammar rules for LR parser, which is currently one of the most popular analyzers, using a simple state transition dialogue model, which is easily derived from a dialogue corpus.

The concept is to limit the grammar rules according to the current dialogue state. The syntactic types of the next user utterance is constrained by the previous sequence of the syntactic types of the utterances. Therefore, it is possible to guide the parser to use grammar rules which are associated with the predicted syntactic types. It directly reduces the syntactic perplexity for the parser and improves both accuracy and efficiency.

4.2.3 Automaton Dialogue Model

Dialogue is modeled with a set of automata, state transition models. Each automaton corresponds to a goal supposed in the task. We set a task domain of personal schedule management, and make automata for registration, modification/cancelling, and query of data. Here, some automata can be called by others to realize sub-goals in main stream of a dialogue. For example, registration dialogue calls the automaton for query, by pushing the current state to the stack, and returns when the query is done.

Each automaton represents a dialogue structure with state transitions, which depend only upon the syntactic types of the user utterances such as correction and confirmation. The state transition is non-deterministic so that multiple types can be allowed.

The structure of automaton is acquired by analyzing real dialogue data labeled with syntactic types. In this experiment, this process is done by human analysis. However, it is easily done automatically by introducing a probabilistic model similar to HMM or N-gram[63]. The obtained automata for the present task domain are shown in Figure 4.8.

Then, for each state of the dialogue automaton, we analyze the acceptable user utterances and describe its syntactic rules. More specifically, each rewriting rule of the grammar is indexed with the state or syntactic type number. Therefore, every time a user utterance is made, its syntactic prediction is provided as a set of grammar rules based on the current state of the dialogue model. Speech parser uses the limited rules, which is expected to have less syntactic perplexity than the whole grammar covering all utterances. Concretely, a dedicated LR parser runs for each dialogue state.

Thus, the overall procedure to construct the dialogue model and the prediction mechanism is as follows:

1. For a training corpus, label the dialogues with their goals and label the utterances with their syntactic types.
2. For each goal, construct a automaton which accepts the sequences of the syntactic types of the utterances.
3. For each state of the automaton, specify the grammar rules which accept the utterances of the possible syntactic types.

As recognition of a utterance is performed with the grammar rules given by the current dialogue state, its syntactic type is identified and a new state of the automaton is specified.

4.2.4 Experimental Evaluation

We have evaluated this method on speaker-independent continuous speech recognition for a spoken dialogue system, named 'secretary system'. The used speech samples are 50 kinds of sentences, all of which are uttered by 8 male speakers. As in Figure 4.8, we set 3 automata and the net number of their states is 12. Among them, 9 states are for user utterances.

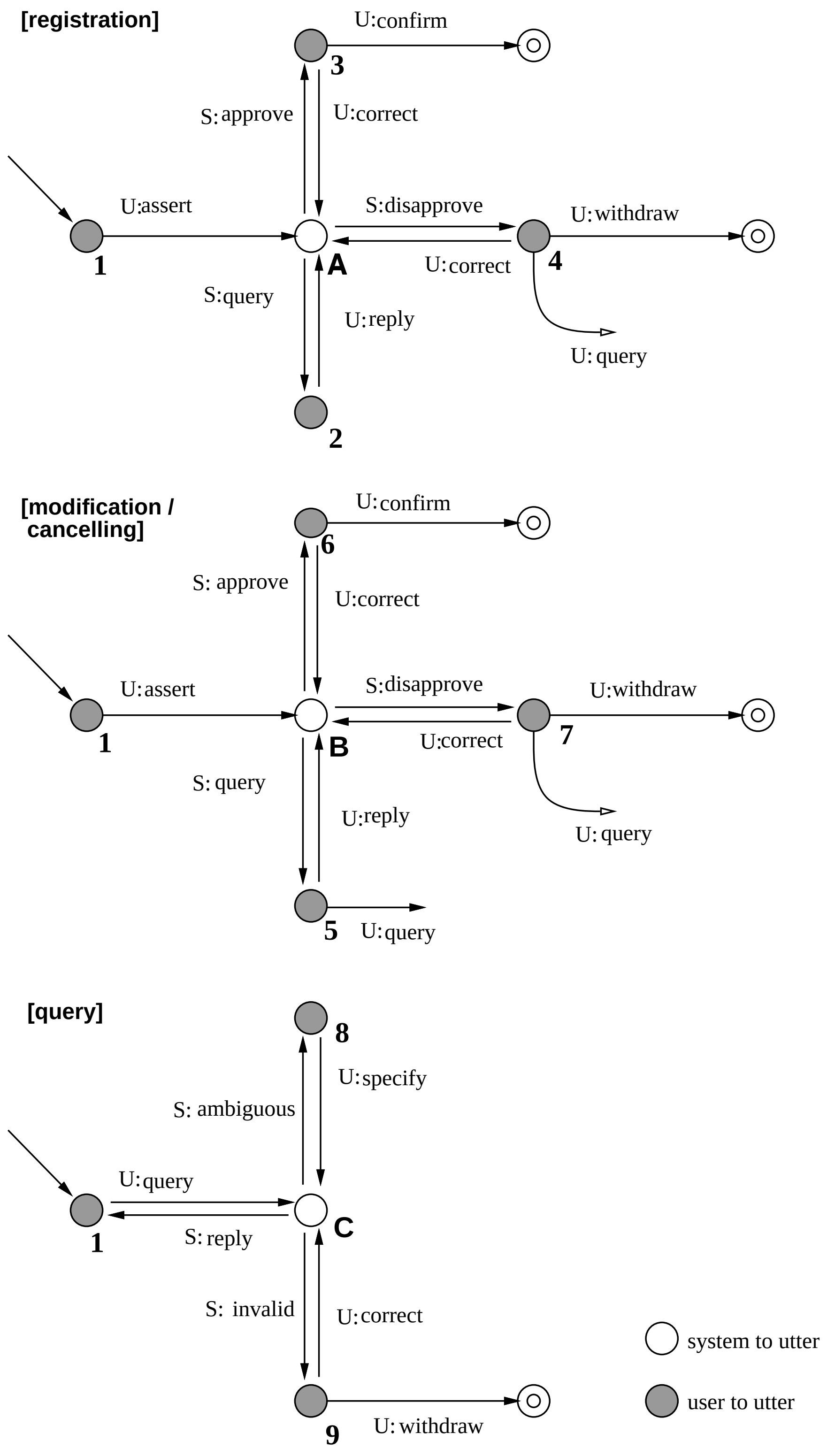

Figure 4.8: Automaton dialogue models

Table 4.5: Effect of dialogue-level prediction for speech parsing (grammar G0)

(Grammar G0)

		word perplex.	sent. rec.	sent. com.	hypo. gene.	hypo pop
st.1	state	35.4	37.9	55.8	4299	51
	common	37.7	34.4	50.4	4893	55
st.2	state	28.2	68.8	75.0	1575	27
	common	37.7	50.0	56.3	2205	26
st.3,6	state	21.7	82.5	85.0	1126	26
	common	37.7	77.5	77.5	1892	22
st.4,7	state	30.9	50.0	55.0	2473	42
	common	37.7	43.8	51.3	3529	39
st.5	state	33.3	50.0	83.3	2256	29
	common	37.7	41.7	83.3	2597	30
st.8	state	28.8	-	-	-	-
	common	37.7	-	-	-	-
st.9	state	25.7	62.5	75.0	1270	23
	common	37.7	43.8	56.3	1734	23
aver.	state	32.3	47.8	61.8	3264	43
	common	37.7	42.0	55.8	3948	44

upper row: state-dependent grammar (with dialogue-level knowledge)
lower row: common grammar (without dialogue-level knowledge)

We first described a common grammar which accepts all the utterances through the dialogue, and then assign state numbers to each rewriting rule to make state-dependent grammars. Here, we used 2 kinds of grammars, which are different in terms of perplexity and coverage. Grammar G0 allows a user more flexible utterances than grammar G1. The word perplexities of G0 and G1 are 37.7 and 17.2 respectively.

Speech recognition is done with pair-wise discriminant discrete HMM. LR parsing is based on A* trellis search with word-pair constraint as heuristics, which gets the 10-best sentence hypotheses.

At first, we evaluated syntactic perplexity of the dialogue state-dependent grammar. The second column of Table 4.5,4.6 lists the word perplexity of the grammar at each dialogue state. The state numbers are referred in Figure 4.8. For state 1, the initial state, dialogue-level prediction does not work. And state 4, 7 and 5 may branch to automaton of query, thus prediction is not effective. As for other states, the perplexity gets half to two third.

Table 4.6: Effect of dialogue-level prediction for speech parsing (grammar G1)

(Grammar G1)

		word perplex.	sent. rec.	sent. com.	hypo gene.	hypo pop
st.1	state	14.4	65.2	70.1	3188	110
	common	17.2	62.5	67.0	3756	115
st.2	state	11.9	81.3	81.3	440	31
	common	17.2	62.5	62.5	2156	48
st.3,6	state	13.7	82.5	85.0	843	44
	common	17.2	85.0	87.5	3241	88
st.4,7	state	16.9	66.3	68.8	1881	68
	common	17.2	66.3	68.8	3338	72
st.5	state	13.8	58.3	91.7	2190	96
	common	17.2	50.0	83.3	2968	85
st.8	state	13.1	-	-	-	-
	common	17.2	-	-	-	-
st.9	state	13.3	68.8	75.0	684	22
	common	17.2	68.8	75.0	1841	42
aver.	state	14.7	67.5	73.3	2422	88
	common	17.2	65.0	70.5	3433	96

upper row: state-dependent grammar (with dialogue-level knowledge)
lower row: common grammar (without dialogue-level knowledge)

Then, we made recognition experiments. Here we assumed for each sentence that the past utterances in the dialogue are correctly understood and the proper dialogue state is given. However, in most cases, recognition errors cause to remain at the current state, which demands the user a re-utterance, and do not transit to improper states. The third and the fourth columns in Table 4.5,4.6 show sentence recognition rates and sentence comprehension rates, respectively. The sentence recognition rate is a ratio of the samples whose word sequences are completely matched. In the sentence comprehension rate, we count as correct if all the keywords which constitute the meaning for the task are completely recognized. The upper row of each slot shows a rate with the state-dependent grammar, and the lower row shows a rate with the common grammar, G0 or G1 itself. The accuracy improves at the states whose perplexities get smaller. It is confirmed that dialogue-level prediction works effectively. However, it is because more than half samples belong to the initial state, namely the first utterance of dialogues, that the average rate for all the samples did not improve so much.

We also investigated the amount of computation with the number of hypotheses tested by the speech parser. The fifth and the sixth columns in Table 4.5,4.6 show the number of generated hypotheses and the number of popped hypotheses, respectively. As dialogue-level constraint limits the search space for the speech parser, the number of hypotheses is reduced. Especially for grammar G1, the dialogue-level knowledge is effective to computation more than to accuracy as its baseline perplexity is smaller.

4.2.5 Conclusion

We have proposed to incorporate dialogue-level knowledge to speech parsing stage. To realize this, a simple state transition model is adopted, which can be easily constructed with a dialogue corpus. The automaton states are directly associated with the grammar rules, which are used by LR parser. This mechanism is straight-forward and needs no overhead computation to make prediction. The dedicated grammar for each state reduced the word perplexity, thus improved the recognition rates and reduced the number of generated hypotheses.

In this paper, we focus on the use of syntactic types characterized by the dialogue structures. We are planning to model more but simple knowledge sources on topics and focus of the dialogue. We are also studying methods to train the dialogue model automatically given a corpus.

4.3 Cooperating with Semantic Analyzer

4.3.1 Introduction

For spoken language understanding or dialogue understanding, a speech recognizer works with a natural language analyzer. It is desirable that they are tightly integrated to share the knowledge sources and limit the search space.

In the conventional model, the recognizer works as a filter for the following language processor, with its own knowledge sources or constraints such as a syntax or a stochastic model. If the filter is loose, its output result may be non-sense sentences. Even if N-best candidates are passed, they all tend to be syntactically or semantically same with one or two words replacements[16], thus structural errors are not always corrected. Therefore, integrating semantic or conceptual-level knowledge sources into the speech recognition stage is significant. Even when we adopt N-best strategy, N-best candidates are to be filtered or merged with semantic-level.

On the other hand, if we try to tune up the syntactic filter of the recognizer to embed all knowledge including semantics and pragmatics, it would be highly hard work and lose robustness against a variety of spontaneous utterances.

A speech understanding system has to accept ill-formed but semantically correct sentences. But when we concentrate on the recognizer of surface word sequences, it will accept either only well-formed or non-sense. We propose, therefore, not direct transplantation of the semantic knowledge into the grammar of the speech parser, but exchanging the knowledge between the speech parser and the semantic parser.

The key to realize this model is the mechanism of incremental semantic analysis and interactive semantic validation. We realize the semantic analyzer with a semantic network.

4.3.2 Incremental Semantic Analysis with Semantic Network

Semantic Network

To realize flexible and robust parsing, we adopt network representation for the semantic analyzer. Unification-type grammar is powerful with strict constraints, but it is hard to relax constraints for noisy inputs. Thus it is not adequate for speech understanding due to the lack of robustness. Template[64] or semantic case-frame[65],[66] based approach is more robust as it can construct semantic representations with fragments of sentences. Therefore, it can be applied to ill-formed utterances. However, it is not easy to manually enumerate possible templates which will drastically increase according to the task size. A semantic network represents the relation of words and their concepts, quite compactly. Taking the paths of arbitrary nodes, a semantic representation is obtained.

The semantic network we adopt is layer structured and loop-free. The lower level represents the lexical knowledge, and the terminals or leaf nodes correspond to keywords that construct the meaning of sentences. The upper level represents the semantic and the pragmatic knowledge, and the top nodes are cores of the semantic types of sentences. The arcs between nodes have several types of attributes such as *is_a* and *instance_of*. A part of the network is shown in Figure 4.9.

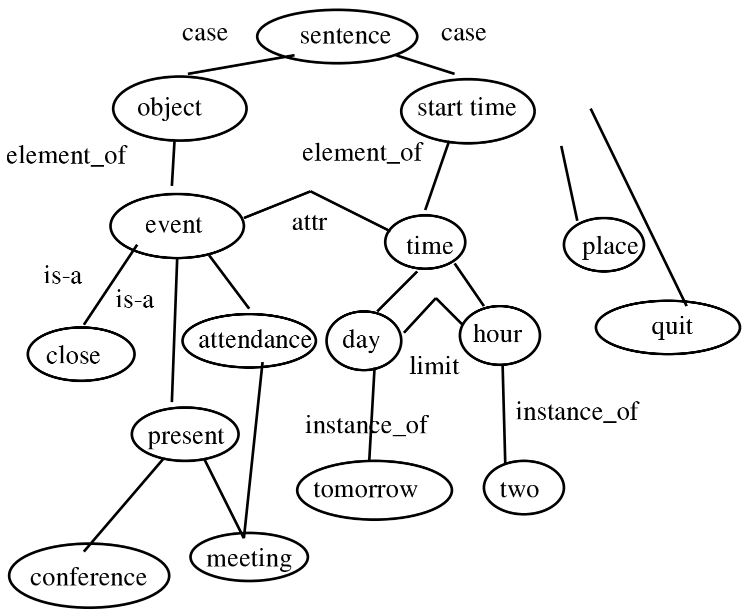

Figure 4.9: A part of semantic network

Incremental Semantic Analysis

The basic strategy to network parsing is classified into two types: marker passing[67] and analogue spreading activation[68]. Here we adopt the former strategy as analogue activation is difficult to formulate and estimate parameters[69],[70].

The semantic representation for a given input is constructed by traversing the paths between activated nodes. When a word is recognized or spotted, the corresponding node of the network is activated. An arbitrary path between two nodes makes a partial semantic representation.

For example, when we recognize 'tomorrow' and 'meeting', then following two paths are found in the network shown in Figure 4.9.

(a) tomorrow $\rightarrow$ day $\rightarrow$ time $\rightarrow$ start_time $\rightarrow$ sentence
$\leftarrow$ object $\leftarrow$ event $\leftarrow$ present $\leftarrow$ meeting

(b) tomorrow $\rightarrow$ day $\rightarrow$ time $\leftrightarrow$ event $\leftarrow$ present $\leftarrow$ meeting

They are transformed to the following semantic representations, respectively.

(a) [assert, [*start_time*, [*time*,tomorrow]], [*event*, [*present*,meeting]]]
(b) [assert, [*event*, [*time*,tomorrow], [*present*,meeting]]]

If necessary, syntactic dependency is checked by referring functional words.

As following words are obtained, new paths with neighboring words are generated and semantic representations are extended. Here semantic consistency is verified and contradictory hypotheses are not extended. A sentence is completed if a semantic representation is fulfilled and the whole input is parsed. Words are picked up so that they cover the whole parts.

The parsing algorithm is described below.

1. Get a word. If it completes a sentence, then finish.
2. Get another word.
3. Traverse paths with neighboring words, and extend semantic representations.
4. If it completes a sentence, then finish. Otherwise go to step 2.

This mechanism realizes robust parsing against minor recognition errors or ill-formed utterances.

Notice that the semantic representation is constructed incrementally as every word is obtained. This incremental semantic analysis is the key to interact with the speech recognizer.

4.3.3 Interactive Semantic Verification Algorithm

Then, we formulate an algorithm of interactive semantic verification. Here, the semantic analyzer works to verify the hypotheses that are generated based on syntactic prediction by the speech recognizer.

For this purpose, we add the semantic checking nodes to the semantic network. It supervises the semantic consistency and alarms on contradictions in various attributes such as place and time. For example, 'traveling' implies going to a distant place unlike home or office, thus 'the destination of the travel is my lab.' is inconsistent. This kind of nodes are made and connected to the related nodes for each attribute. An example of semantic checking nodes is shown in Figure 4.10. When some leaf node is activated, the attribute markers are passed to the semantic checking nodes. If a semantic checking node detects any contradiction, the hypothesis is invalidated and abandoned immediately.

Namely, the semantic verification is featured to the incremental semantic parsing algorithm. The step 3 is modified as follows.

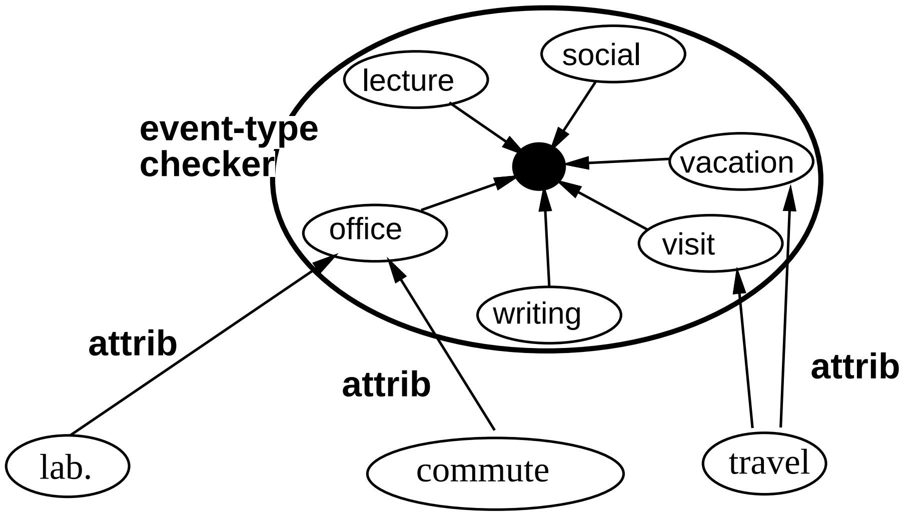

Figure 4.10: An example of semantic checking nodes

3-a Traverse paths with neighboring words, and extend semantic representations.

3-b Pass the attribute markers to the semantic checking nodes.

3-c If a semantic checking node detects any contradiction, invalidate the hypothesis.

This incremental semantic verification reduces perplexity at the recognition stage as a result. We can evaluate this effect with the test-set perplexity measure.

However, it would be computationally expensive to call the semantic analyzer for all the hypothesis extensions. In our implementation, the semantic analyzer is called every time a hypotheses is popped from the hypothesis stack, in order to verify it. Namely, sentence hypotheses are generated based on syntax, but one is passed to semantic verification only when it gets to the stack-top or best at a time. So this interaction mechanism works well with a stack decoder rather than beam search. The mechanism can be implemented to stack decoding search by storing the status of the network, concretely activated nodes, for each hypothesis as well as its LR parsing state.

4.3.4 Experimental Evaluation

We evaluated how interactive semantic analysis contributes to speech recognition based on LR parser. We incorporate interactive semantic verification into A* search based on stack decoder.

The results are shown in Table 4.7. As well as the word and the sentence recognition rates, the semantic accuracy is also evaluated. The semantic accuracy is the ratio of the

Table 4.7: Effect of interactive semantic verification

	word (top-10)	sent. (top-10)	semantic
G0	80.4 (92.3)	42.0 (71.5)	60.5
G0 + semantic	80.9 (91.7)	45.3 (74.3)	63.5
prob-G0	82.6 (93.8)	46.5 (76.5)	65.5
prob-G0 + semantic	83.7 (93.8)	50.3 (78.8)	69.0
G1	86.2 (93.9)	65.0 (89.3)	71.3
G1 + semantic	86.0 (93.3)	65.8 (90.3)	72.3
prob-G1	88.2 (95.2)	67.0 (90.0)	73.5
prob-G1 + semantic	88.1 (94.7)	67.8 (90.8)	74.5

samples whose semantic representations are correctly obtained. The second column (word) and the third column (sent.) list the word and the sentence recognition rate, respectively. The last column (semantic) shows the semantic accuracy. We compared with normal LR parsing. In this case, we first applied LR parser to get 10-best sentence candidates, and performed semantic analysis in turn, until a complete semantic representation is obtained.

In any cases, interaction with semantic analyzer improves the accuracy. Notice that the same semantic knowledge source is applied even to the grammar-only case to get semantic representations. The difference is timing and the way of applying them. This means that interacting with semantic analyzer at the recognition stage is effective.

However, the improvement gets smaller as the baseline recognition rate gets higher. This is because a stronger grammar such as G1 embeds the semantic knowledge to large extent. But it is still hard to embed all the semantic constraint such as attributes of words in the LR grammar. Even if possible, such a fixed grammar is not adequate for dealing with a variety of sentences. The coupling of a loose grammar G0 and the interactive semantic verification accepts more flexible utterances. In other words, it realizes more robust understanding.

4.3.5 Overall Evaluation of Knowledge Sources

Finally, we make evaluation of all the knowledge sources described so far. They are probabilistic LR grammar, dialogue-level prediction and interactive semantic verification. They are incorporated in the framework of A* search whose heuristics is word-pair or word bigram. It is illustrated in Figure 4.1. The baseline is LR parsing that contains the syntactic and lexical knowledge, as discussed in Section 4.1.

In order to see the effect of each knowledge source, we incorporated one by one. Table 4.8 and Table 4.9 show the results for grammar G0 and G1 respectively, with the word recognition rate (word), the sentence recognition rate (sent.) and the semantic accuracy (semantic). The strength of each constraint is measured with the test-set perplexity.

It is clearly seen that integrating more constraints brings better accuracy. Notice again that, for all experiments including the grammar-only case, the same semantic and dialogue-level constraints are applied to extract the meaning of the utterances. The difference is that they are applied after getting the $N(=10)$-best sentence candidates. This means that incorporating higher-level knowledge sources into speech parsing stage is significant. The semantic and dialogue-level knowledge sources work quite independently. Namely, the improvement of the accuracy by the both constraints is almost the sum of those by the respective ones.

The full integration of knowledge sources improved the sentence accuracy by 10% with respect to grammar G0. This coupling of a loose grammar and the semantic and dialogue-level constraints accepts ill-formed sentences while filtering out non-sense ones. As for grammar G1, the improvement is not so significant, as the semantic knowledge is partly embedded in the grammar itself.

From the computational viewpoint, this strategy turned out not so expensive. The language processing time is much less than HMM trellis computation. Moreover, the semantic analyzer eliminates vain hypotheses at an early stage, and the dialogue-level constraint is applied to limit the grammar of the parser before whole sentence recognition. Actually, the extra CPU time of the full integration was 8~11% of that of the baseline parsing.

Table 4.8: Effect of integrating linguistic knowledge into grammar G0

	test-set perplex.	word (top-10)	sent. (top-10)	semantic
G0	33.7	80.4 (92.3)	42.0 (71.5)	60.5
G0 + semantic	31.5	80.9 (91.7)	45.3 (74.3)	63.5
G0 + dialogue	28.5	83.6 (93.9)	47.8 (76.0)	64.8
G0 + semantic + dialogue	26.5	84.5 (93.2)	51.5 (77.5)	67.3
prob-G0	28.0	82.6 (93.8)	46.5 (76.5)	65.5
prob-G0 + semantic	26.6	83.7 (93.8)	50.3 (78.8)	69.0
prob-G0 + dialogue	25.0	84.8 (94.5)	51.0 (79.0)	67.8
prob-G0 + semantic + dialogue	23.6	85.7 (94.1)	54.0 (80.8)	70.3

Table 4.9: Effect of integrating linguistic knowledge into grammar G1

	test-set perplex.	word (top-10)	sent. (top-10)	semantic
G1	17.4	86.2 (93.9)	65.0 (89.3)	71.3
G1 + semantic	16.7	86.0 (93.3)	65.8 (90.3)	72.3
G1 + dialogue	16.7	87.9 (95.1)	67.5 (90.3)	72.8
G1 + semantic + dialogue	16.0	88.2 (95.0)	68.5 (91.5)	74.0
prob-G1	16.5	88.2 (95.2)	67.0 (90.0)	73.5
prob-G1 + semantic	15.9	88.1 (94.7)	67.8 (90.8)	74.5
prob-G1 + dialogue	16.2	89.3 (96.3)	68.5 (90.8)	73.0
prob-G1 + semantic + dialogue	15.5	89.3 (96.0)	69.0 (91.5)	75.5

4.3.6 Conclusion

We have presented an interactive semantic analysis method to verify partial sentence hypotheses. Incremental semantic analysis is implemented with a semantic network. It concurrently works using the same stack decoder with the LR parser. It is shown that the mechanism improves the accuracy.

Finally, all the available linguistic knowledge sources are integrated and evaluated in the framework of the single A* search with word-pair or word bigram as heuristics. It is shown that integrating the semantic and dialogue-level knowledge into the speech recognition stage achieves better accuracy than the conventional N-best strategy that applies them after getting the N-best sentence candidates.

Chapter 5

Word and Phrase Spotting with Heuristic Language Model

In a spoken language system or a spoken dialogue system, it is significant to deal with natural spontaneous speech including ill-formed sentences. The ill-formedness such as fillers, hesitations and unknown words is of variety, and it is hard to model and describe all the obstructing phenomena.

When the task domain is rather limited, we can mostly make senses of utterances by putting attention to keywords and discarding the rest detailed parts. In a spoken dialogue system, it is also possible to make clear the unrecognized parts through the following dialogue. Therefore, spotting-based approach that extracts only recognizable parts and skips the rests is attractive[71].

The basis of this approach and the key to its success is keyword spotting. Although it has been studied a lot, the spotting strategy has not succeeded so well especially at a large-vocabulary task. The reason is spotting itself fails to obtain enough detection accuracy, and causes too many false alarms to deal with in the following processing.

To be more specific, we contend the following issues in the conventional spotting approaches.

1. No use of language models at the spotting stage
2. Use of words as the spotting unit
3. Deterministic segmentation at the spotting stage

To overcome the above defects, we adopt heuristic and progressive search strategy. It applies multiple-level constraints sequentially in the order of their strength.

1. Simple language model such as word/syllable bigram for the whole utterance.

2. Local phrase syntax at the spotting stage

3. Inter-phrase semantic constraint for sentence understanding

The concept is fundamentally same as that presented in Chapter 4 which is illustrated in Figure 4.2.

The important point is we store the intermediate results of HMM trellises at each step except the last one. It is provided to the following step as the heuristics which realizes admissible and efficient search, except that the first pass is Viterbi search. The strategy progressively limits the search space and performs efficient search, obtaining good heuristics. By retaining the intermediate HMM trellises, it gets possible to compute the globally optimal Viterbi score. By choosing such constraint at every step as subset of that for the next step, the obtained heuristics gets A*-admissible, which guarantees the next search to find the optimal hypothesis.

In the following sections, we discuss each issue in the above order: the heuristic language model for spotting in Section 5.1, the phrase syntax as a spotting unit in Section 5.2, and the inter-phrase semantic constraint in Section 5.3.

5.1 Word Spotting with Heuristic Language Model

5.1.1 Introduction

Word spotting algorithms have been studied a lot. However, almost all of them focus on acoustic modeling of the spotted words themselves. In this section, we propose to incorporate a low-level language model to sentences in which keywords are contained, and to spot keywords which satisfy the constraint with high probabilities. In this formalism, we define the word spotting problem as finding the most probable (sentence) hypothesis for a whole utterance which contains the word.

The most important issue for this approach is to construct a language model which will be effective constraint but does not hamper the flexibility of user utterances. We examine several language models and present experimental evaluations.

5.1.2 Heuristic Word Spotting

Our goal here is to spot all the significant words which contribute to sentence understanding in a given task domain. In this kind of multiple word spotting in a sentence utterance, we can assume that an input contains some word to be spotted as well as other words. Namely, an input is a sequence of spotted words, other (unknown) words, filled and silent pauses. This task is rather different from word spotting in continuously recorded material or noisy environment. Here, we know the input is a sort of language though it is very ill-formed. Therefore, it is desirable to model the language and incorporate this knowledge into the spotting phase.

The conventional study on word spotting mainly focuses on how to construct the acoustic model for spotted words and does not care the rest parts. In the spotting strategy, the word model is matched assuming that every time-frame be a starting point or an end point of the word. But it is difficult to compare Viterbi scores that are different in length (scoring problem). It is also hard to precisely identify starting points and end points that give the maximum score (segmentation problem), without investigating the neighboring parts. This sort of bottom-up matching is always annoyed with the local noise or similarity, which will cause unnatural false alarms.

Here we propose to put linguistic constraint on the whole input, and evaluate a hypothesis that contains the target word and satisfies the constraint. The evaluation function for a hypothesis that the input contains word w in time $t_1 \sim t_2$ is defined as the sum of the score $g(w, t_1, t_2)$ for the word itself and the score $h(w, t_1, t_2)$ for the rest part $(1 \sim t_1, t_2 \sim T)$. The spotting model is illustrated in Figure 5.1.

$$f(w, t_1, t_2) = g(w, t_1, t_2) + h(w, t_1, t_2) \tag{5.1}$$

The heuristic score $h(w, t_1, t_2)$ that the remaining part makes a plausible sentence in a task is ignored as zero in conventional strategies. Since we incorporate the heuristic knowledge on language, we call this strategy heuristic word spotting. The word spotting is formulated as finding the N-best hypotheses based on this evaluation function $f(w, t_1, t_2)$.

The heuristic score is divided into the preceding part $(1 \sim t_1)$ and the following part $(t_2 \sim T)$ of the word.

$$h(w, t_1, t_2) = h_l(w, 1, t_1) + h_r(w, t_2, T) \tag{5.2}$$

We call them left-context heuristics $h_l(w, 1, t_1)$ and right-context heuristics $h_r(w, t_2, T)$, respectively.

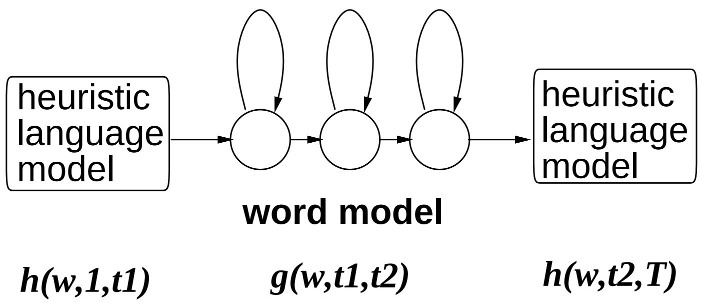

Figure 5.1: Spotting model with a heuristic language model

This formulation solves the problems of scoring and segmentation which arise in length-free matching, as it judges after scanning the whole input. A similar approach is based on the use of a garbage model[72]. The garbage model represents unknown words and filled pauses. It is usually trained with the whole part of speech samples. The garbage model, a complete wild card, will accepts any inputs. However, it does not work as constraint for inputs, since it is not based on linguistic knowledge. The linguistic constraint must be embedded with other forms such as automaton-type grammar[73],[74],[75] which performs parsing, not spotting. Furthermore, the garbage model is completely a black box. Its training is unstable and its scoring is not unreliable, which is confirmed only after trials and errors.

Our strategy uses a language model that is not a strict grammar but constrains inputs to be plausible sentences in a task. It will substantially reduce the perplexity of inputs and suppress unnatural false alarms. Moreover, since the heuristic model is constructed with the phone model that is also used for the word models, it will provide consistent scores with the word scores. It is also possible to set the threshold for spotting dynamically with the heuristic score.

5.1.3 Spotting Algorithm

Heuristic spotting algorithm consists of two phases: heuristics computation and spotting itself.

First of all, the heuristic model is applied to the whole part of an input speech. Although the same model is used to approximate the left part and the right part of the spotted word, Viterbi scores for the two must be independently computed. Therefore, we apply the model both left-to-right and right-to-left, and stores the respective generated

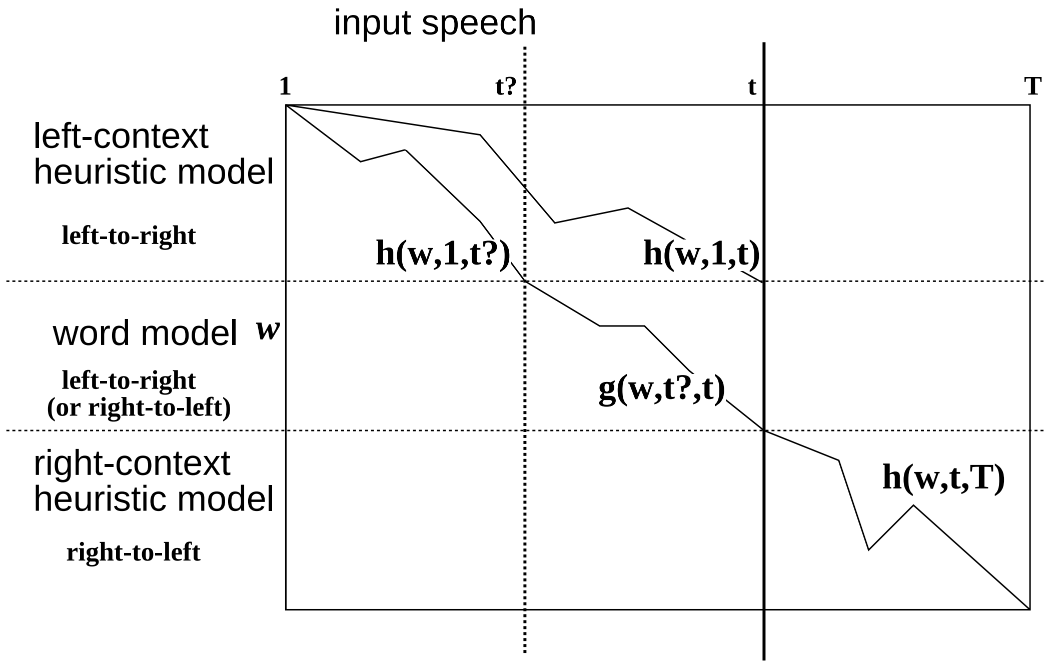
input speech
1
t?
t
T
left-context heuristic model
left-to-right
word model w
left-to-right (or right-to-left)
right-context heuristic model
right-to-left
h(w,1,t?)
h(w,1,t)
g(w,t?,t)
h(w,t,T)

Figure 5.2: Trellis concatenation of word model and heuristic language model

trellises. The evaluation function $f(w, t_1, t_2)$ is obtained by concatenating three trellises as shown in Figure 5.2. This evaluation is performed for each word w at each time-frame t, and those candidates that exceeds a threshold are output as spotted candidates. The algorithm is formulated as follows.

1. The heuristic model is applied left-to-right, and the left-context heuristic trellis $h_l(w_*, t_*, T)$ is computed for every word w_* at every time-frame t_*, with Viterbi algorithm.

2. The heuristic model is applied right-to-left, and the right-context heuristic trellis $h_r(w_*, t_*, T)$ is computed for every word w_* at every time-frame t_*, with Viterbi algorithm.

3. For each word to be spotted w, its model is evaluated. Perform the following for each time-frame t.

   (a) The initial state is connected to the left-context heuristic model. Namely, the self-loop of the initial state is compared with the transition from the end of the heuristic model. This incorporates the left-context heuristics $h_l(w, 1, t)$

   (b) For the following states of the word model, Viterbi computation is performed.

(c) The final state is followed by the right-context heuristic model. Namely, the the score of the right-context heuristic trellis $h_r(w, t, T)$ is added to compute the evaluation function $f(w, t_?, t)$.

(d) If its value is over the threshold, then the word w is output as a candidate. The end point is t, and its corresponding starting point $t_?$ is back-traced from t. If this candidate overrides the previously output one that is the same word in the overlapped segment, the best one is chosen and the rest is deleted.

The last step generates, for each time-frame t, a hypothesis that word w ends at t. Actually, starting time-frame $t_?$ of the word is identified by Viterbi alignment only after the word is spotted. The left-context heuristic score $h_l(w, 1, t_1)$ is embedded as the word trellis is generated, while the right-context heuristic score $h_r(w, t_2, T)$ is explicitly added to complete the evaluation function.

When we use a heuristic language model that includes the words to be spotted such as word-concatenation model, the score of the word part is obtained in heuristics computation. Thus, we can omit either step 3-(a)$\sim$(b), or step 3-(b)$\sim$(c).

Fundamentally, the overall procedure becomes three pass algorithm on the input including the spotting phase itself. However, it is possible to perform the two heuristics computations (step 1 and 2) in parallel. Or, when we perform step 3 in the right-to-left direction, it is possible to perform step 2 and step 3 in parallel and frame-synchronous. This is most efficient as step 1 can be processed in real time.

We can seek more efficient search algorithm based on this formula such as best-first search. But here we realize a full Viterbi search, as the word hypothesis extension is small enough.

Threshold for Spotting

The threshold for spotting can be set dynamically based on the score of the optimal path with the heuristic language model $P_{max} = h(w_*, 1, T)$. Generally, an acoustic score such as Viterbi score is affected by input conditions, and it is hard to decide the threshold based on the absolute value. Since the score P_{max} is optimal for the whole utterance, it gives the upper bound of the evaluation function $f(w, t_1, t_2)$. Thus, we can normalize the acoustic scores by choosing a relative value of the heuristic score as the threshold.

Scoring of Candidates

The evaluation function $f(w, t_1, t_2)$ reflects the whole input. Its intuitive meaning is spotting cannot be done without considering neighboring contexts. However, it might be inadequate to use this evaluation function for scoring the word itself, since it is the score of a hypothesis that the word exists.

From this viewpoint, the score of the word itself can be modified as $g(w, t_1, t_2)$ only. It is computed with Viterbi alingment for the word part after the word spotting hypothesis is adopted. In this case, some normalization on its length is necessary. Here, we just divide logarithm scaled Viterbi score g with the length $(t_2 - t_1 + 1)$.

5.1.4 Language Modeling for Heuristics

The key of the spotting strategy is language modeling for heuristics that will be effective constraint and robust against a variety of user utterances. Several models are examined.

Syllable-Concatenation Model

This model represents just a parallel concatenation of the possible syllables. It is self-looping of syllable models that are possible in the language. In Japanese, there are about 100 CV (Consonant-Vowel) syllables. Thus, this model can accept any Japanese sentences. But it is the loosest constraint without lexical knowledge. A part of the model is shown in Figure 5.3.

It is desirable to use syllable bigram since it characterizes the general language better. Bigram or a transition probability of the syllables is assigned to each concatenation. It is known that bigram reduces perplexity of the language[76]. It is estimated with a large corpus.

Word-Concatenation Model

This model approximates an input utterance with a sequence of known words using large vocabulary lexical knowledge[77],[75]. It is a parallel combination of all the possible word models. In addition to the keywords to be spotted, functional words are added to the lexicon. An example of the model is shown in Figure 5.4.

Since it limits the vocabulary as in conventional parsers, the score for an input including unknown words or filled pauses is not guaranteed as correct. To cope with spontaneous

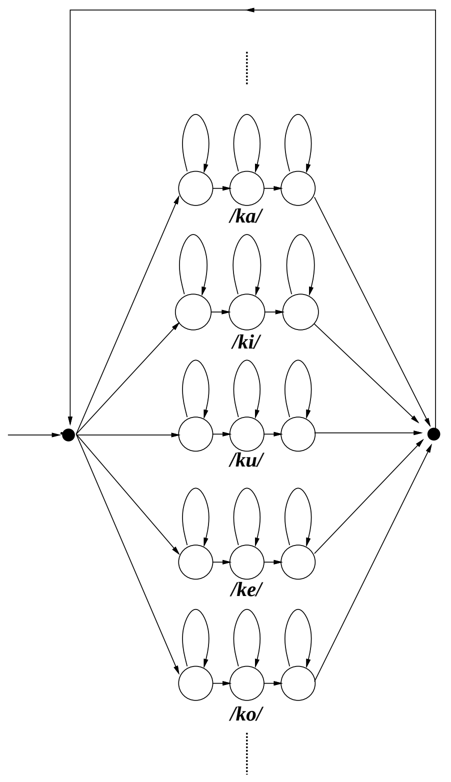

Figure 5.3: Syllable-concatenation model

utterances, models of filled pauses can also be added.

Word-Pair Model

In addition to the lexical knowledge, this model constrains the connections of the words. Only the syntactically admissible pairs of the words are connected. An example of the model is shown in Figure 5.5.

This constraint will improve the spotting accuracy, especially concerning short words whose phone sequences are similar to other words locally, for example, '*kyou* (today)' and 'Tokyo'. But syntactic constraint on the word connections is also often violated in Japanese conversational speech.

In this model, too, filled pauses can be incorporated.

Word-Syllable Hybrid Model

Generally, more constraint will improve the spotting accuracy but loses robustness. The ideal mechanism applies strong constraint for an input that satisfies it, and relaxes constraints when an input violates it. Here, we propose a hybrid model of words and syllables.

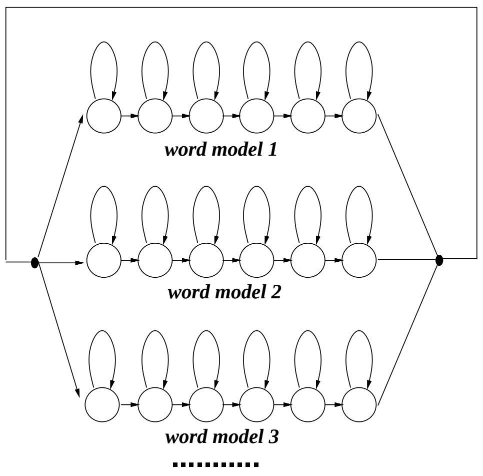

Figure 5.4: Word-concatenation model

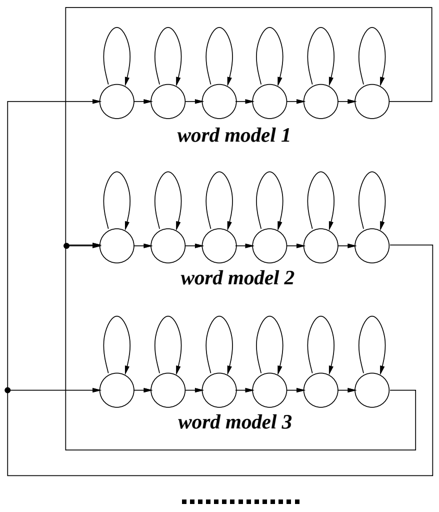

Figure 5.5: Word-pair model

It is a combination of syllable-concatenation and word-concatenation, as shown in Figure 5.6. The part of an input that makes some known word is expected to pass the corresponding word model, and the unexpected part such as unknown words and filled pauses will pass the syllable models.

As both syllable models and word models consist of the same phone model, the probability of passing the syllable models must be lowered relatively, lest the whole input pass the syllable models. It is regarded as a penalty for violation of the lexical constraint.

In this model, too, syllable bigram can be incorporated to the syllable-concatenations in order to improve the model.

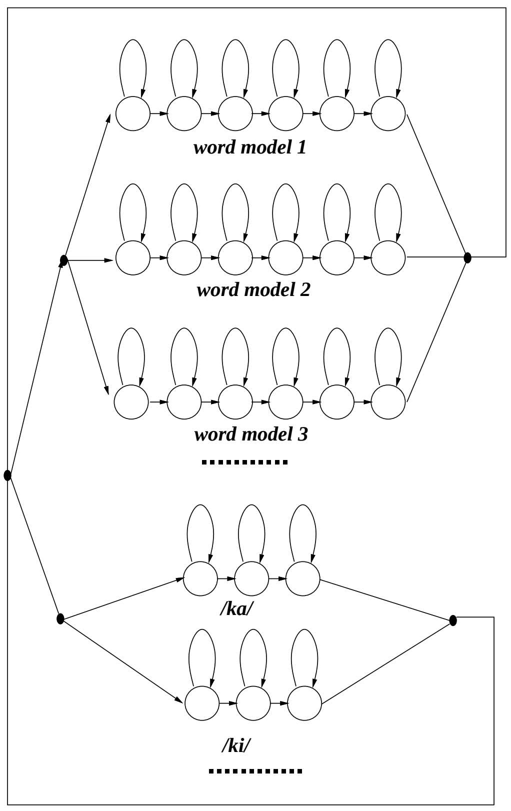

Figure 5.6: Word-syllable hybrid model

5.1.5 Experimental Evaluation

Heuristic language models are constructed as follows.

Syllable-concatenation model consists of 110 Japanese CV (Consonant-Vowel) syllables. In order to estimate syllable bigram, we used dialogue text database of the Acoustic Society of Japan (ASJ)[30]. It is completely independent from our task domain. It has about 120 thousands of syllables. The syllable perplexity with bigram gets 21.6. Here we performed back-off smoothing on the bigram of the syllable pairs that do not appear in the corpus frequently[31].

Word-concatenation model consists of 230 words, including functional words as well as the words to be spotted. Moreover, we construct another word-concatenation model that also includes 6 filled pauses. Word-pair model constrains above word-concatenation models. Word-pair is derived by LR grammar described for continuous speech parsing. The word perplexity with the word-pair is 46.5 without fillers and 52.9. with fillers.

The threshold for spotting is set based on the optimal score with a heuristic language model P_{max}. After the preliminary experiments, we set it to $P_{max} \times 1.05$ on the logarithm scaled Viterbi score, so that enough candidates are obtained. We limit the appearance of one word in a utterance to 5 at maximum.

In order to evaluate if the spotted candidate is correctly located, we performed the Viterbi alignment on the inputs with the correct models beforehand. Correctness of the segmentation is judged by comparing with this alignment. We count as correctly segmented if the midst point of the spotted candidate exists between the starting point and the end point of the correct alignment.

The overall spotting accuracy is evaluated with the number of correct words (A) and incorrect words (FA: False Alarm) in a certain candidates. The ratio of detecting correct words (A) to the number of words to be spotted in the sample (W) is defined as detection rate (A/W). The number of false alarm (FA) is normalized by the product of the vocabulary size of keywords (KW) and total time of the samples (in hour: H), both of which it depends on. We notate this FA/KW/H (=False Alarm per KeyWord per Hour). In our experiment, the total time of 400 samples of grammatical sentences is 1260 sec., namely 0.350 hours, and that of 200 samples of ill-formed sentences is 806 sec., that is 0.224 hours.

When most of the vocabulary are keywords and large part of an utterance is recognized, the word recognition rate can be computed in much the same way as in continuous speech parsing. Here, we investigate the detection accuracy (A/W) in the same candidates as the number of words to be spotted (W), which we notate A/1W. In the large vocabulary word spotting, which is designed for robust speech understanding, we basically use this evaluation measure.

When the set of keywords are limited, only small part of an utterance is to be spotted. Naturally, the detection rate improves when we allow more false alarms. In this case, we have to investigate the detection rate at various FA/KW/H values. The graph of the spotting rate A/W versus the false alarm rate FA/KW/H is called ROC (Receiver-Operating-Characteristic) curve. As a compact measure for the spotting performance, a Figure of Merit (FOM) score is defined as the average detection rate from 0 to 10 FA/KW/H. In the small vocabulary word spotting, we use this measure.

Evaluation with Large Vocabulary

In order to evaluate our methods, we performed large vocabulary word spotting experiments. The words to be spotted are all the noun, verbs, and several functional words

Table 5.1: Result of 219 vocabulary spotting

language model	grammatical		ill-formed	
	A/1W	FA/KW/H	A/1W	FA/KW/H
no heuristics	31.7	15.33	28.4	13.19
garbage	42.6	12.85	35.7	11.84
syllable-conc.	36.6	14.10	42.0	10.68
syllable-bigram	42.4	12.92	48.7	9.46
word-conc.	60.5	8.86	53.0	8.66
word-conc. (fillers)	59.4	9.11	55.8	8.15
word+syllable-conc.	60.2	8.92	55.6	8.17
word+syllable-bigram	60.0	8.98	55.6	8.17
word-pair	74.5	5.71	58.0	8.15
word-pair (fillers)	73.5	5.94	67.0	7.75

to determine the cases of phrases. The vocabulary size is 219. The average number of phonemes per word is 7.83. Phone models are based on discrete HMM with pair-wise symbol classifiers.

As well as the proposed methods, we also applied the conventional spotting algorithm based on length-free matching, which only uses word models and does not assume any language models. If a local peak of the Viterbi score normalized by the frame length exceeds a threshold, then the word is spotted. Moreover, we implemented a garbage model as the background model for reference. It is constructed by taking the average distribution of all the phone models.

The results for grammatical sentences and ill-formed ones are shown in Table 5.1. The table lists the word detection rate (A/1W) and the false alarm rate (FA/KW/H) in the W-best candidates. The relation between the detection rate (A/W) and the number of candidates is shown in Figure 5.7 for grammatical utterances and in Figure 5.8 for ill-formed ones, respectively. The number of candidates is measured as a ratio to W.

These results show that spotting with a heuristic language model is more effective than the conventional matching. The conventional length-free matching spots words based on the local score, thus tends to generate false alarms in such parts that is locally similar to the words to be spotted.

However, syllable-concatenation model does not work well, since it does not give any constraints on language. It just works as syllable alignment of the whole input. Incorporating bigram of the syllables improves the spotting accuracy. It succeeds to model

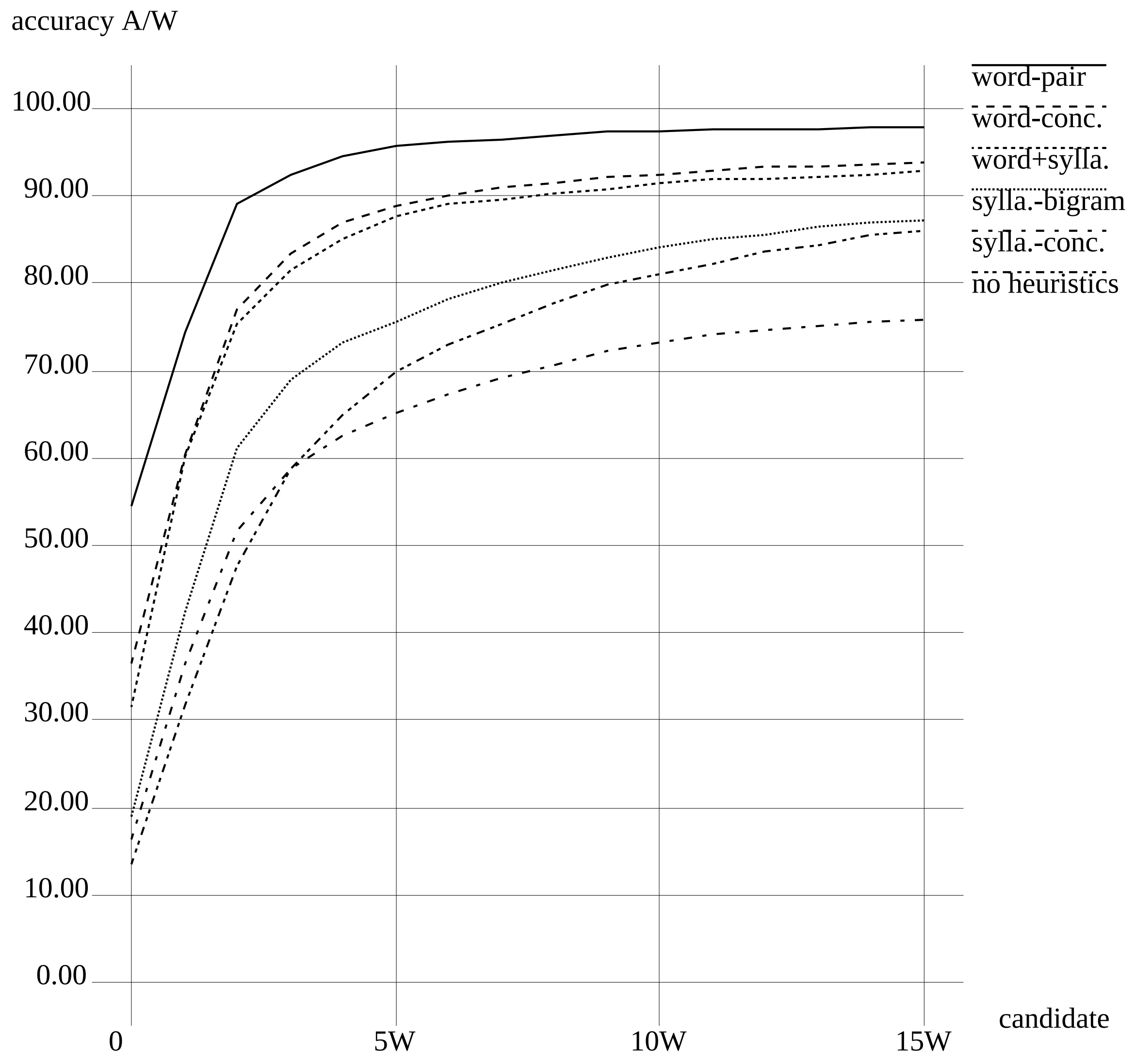

Figure 5.7: Result of 219 vocabulary spotting (grammatical utterances)

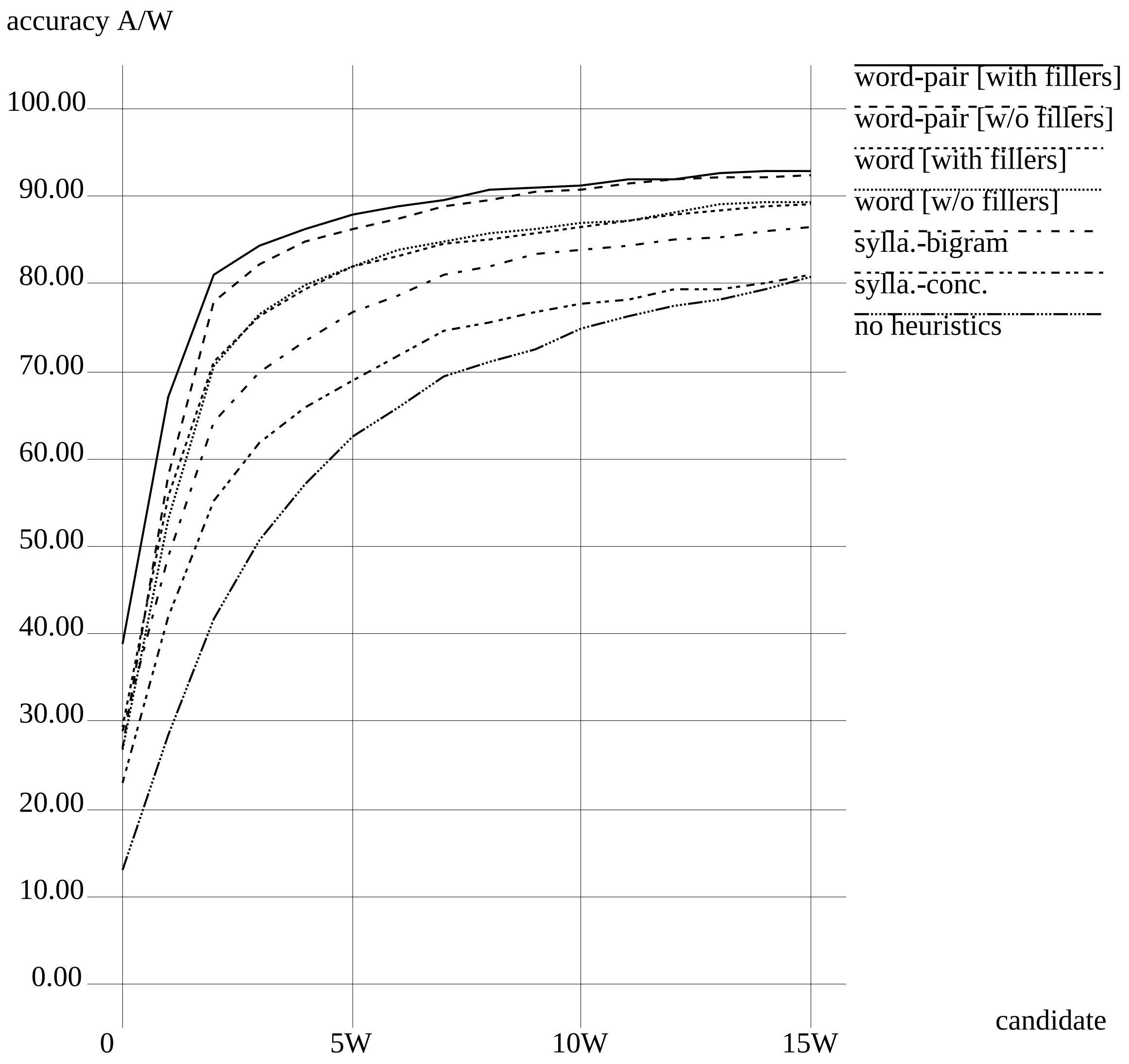

Figure 5.8: Result of 219 vocabulary spotting (ill-formed utterances)

Table 5.2: Comparison of scoring of spotted candidates

language model	grammatical		ill-formed	
	f A/1W	g A/1W	f A/1W	g A/1W
syllable-bigram	42.4	32.7	48.7	29.4
word-conc. (fillers)	59.4	36.6	55.8	28.4
word-pair (fillers)	73.5	46.6	67.0	36.7

general spoken language even if its training text has nothing to do with the task domain.

The garbage model is more effective than the length-free matching, but it gives as weak constraint as the syllable-level models do.

Word-concatenation model works quite effectively. It brought about better accuracy than any syllable-level models. In grammatical utterances where no fillers exist, using fillers in the heuristic model had only bad effect, as it increases perplexity of the model. In ill-formed utterances, the model with fillers was effective, but only a little. In other words, word-concatenation model without fillers works quite well even in ill-formed utterances that violate the lexicon. It is because fillers can be approximated by several short (one-syllable) words such as functional words.

Word-syllable hybrid model also works well, but it got much the same accuracy as word-concatenation model only. Incorporating bigram of the syllables had no effect in this case. It means that word-level constraint is so powerful and robust that syllable-level knowledge makes little sense.

Word-pair model got the best spotting accuracy in the experiments. While syllable-concatenation model allows any sequences, word-pair model does not approximate the rest part of a mis-placed word and prevents it from being spotted. But in ill-formed utterances, the model without fillers is less effective. It means word-pair is not robust against spontaneous utterances where the constraint is not satisfied.

Comparing the results for grammatical sentences and ill-formed sentences, as the heuristic model gets powerful, it works less effectively for ill-formed sentences than for grammatical ones.

Next, we examined evaluation function for the spotted candidates. In the above experiments, we used $f(w, t_1, t_2)$ for scoring. For comparison, then, we used $g(w, t_1, t_2)$ which reflects only the part of the word w. The spotting algorithm is the same, namely based on the evaluation function $f(w, t_1, t_2)$ using a heuristic language model. The difference is

Table 5.3: Result of 26 vocabulary spotting

language model	grammatical		ill-formed	
	FOM	A/1W	FOM	A/1W
no heuristics	17.8	48.3	26.5	49.3
garbage	50.9	73.8	36.9	64.0
syllable-conc.	14.0	37.9	56.0	77.2
syllable-bigram	20.1	49.2	57.5	78.7
word-conc.	54.4	78.8	69.9	86.8
word-conc. (fillers)	50.7	76.2	71.4	88.2
word+syllable-conc.	63.0	76.2	73.0	88.2
word+syllable-bigram	62.5	76.7	73.1	88.2
word-pair	73.6	90.0	62.0	86.0
word-pair (fillers)	74.1	89.6	68.7	89.7

scoring or ranking of the spotted candidates. The results are shown in Table 5.2. In all the cases $f(w, t_1, t_2)$ brings about much better accuracy. It is significant to use heuristics for scoring as well as for spotting itself. Especially, the score $g(w, t_1, t_2)$ is not reliable for the unnatural matching such as staying at one HMM state too long.

Evaluation with Small Vocabulary

For a reference, we performed a spotting experiment with rather small vocabulary. We chose 26 keywords, all of which are concerning the scheduled events in our task domain, for example, meeting, lecture and conference. The average number of phonemes per word is 10.3. The results are shown in Table 5.3. Here, we compute a Figure of Merit (FOM) that is the average detection rate (A/W) at various (0$\sim$10) false alarm rates (FA/KW/H), as well as A/1W in the W-best candidates.

Naturally, the absolute performance gets much better than 219 word spotting, as the vocabulary size drastically decreases. However, much the same tendency is seen with respect to the comparison of the heuristic language models. In this case, word-pair model without fillers is inferior to any word-level models. It makes clear the lack of robustness of word-pair constraint.

Only difference is that syllable-level models for grammatical sentences were so worse. Investigating the erroneous samples, we found many cases that a word w_0 is mis-spotted at the part of word utterance w_1 that has w_0 as prefix or suffix, for example 'meeting (w_0)' spotted in 'joint-meeting (w_1)'. As the syllable model approximates the rest part well or even better than longer correct word w_1, the score of w_0 gets higher than that of w_1. The samples of grammatical sentences has many such cases as for this vocabulary set.

5.1.6 Conclusion

We have proposed a new word spotting strategy, which incorporates a language model as heuristics. It judges the word existence based on the evaluation function which reflects not only the matching score of the word itself but also the plausibility of the rest part as a sentence.

Experimental results show this heuristic word spotting strategy is more effective than the conventional bottom-up matching-based method.

We have also examined which language model is appropriate with respect to the accuracy and robustness, and come to several conclusions. (1) syllable-level knowledge is not sufficient. (2) word-level constraint is very effective and robust against spontaneous speech. (3) word-pair constraint is most powerful but not robust.

5.2 A*-admissible Phrase Spotting Algorithm

5.2.1 Introduction

For robust spoken language understanding, we have studied a word spotting algorithm, especially on the use of a heuristic language model on whole utterances. Though it constrains inputs and suppresses unnatural false alarms, the overall word accuracy is still inferior to that of the continuous speech parsing approach. It is because the word connections are not constrained with syntactic knowledge at all in the spotting approach.

In this section, we propose to use phrase-level syntax at the spotting stage. Since a phrase is uttered at one moment, its syntax is rarely violated even in spontaneous utterances. It will reduce the task perplexity and improve the spotting accuracy. Thus, it is a hybrid of the parsing approach and the spotting approach, whose merits are reflected.

5.2.2 Heuristic Phrase Spotting

In the spotting-based approach, keyword has been commonly used as the unit or target of the spotting. However, word spotting that uses small templates is still annoyed with the local noise or similarity. Using a longer unit is advantageous because it can incorporate more distinguishing information, that is linguistic knowledge. Here we propose to use phrase as the spotting unit.

A phrase consists of a few keywords and functional words, for example, 'from Tokyo', or 'at three o'clock in the afternoon'. Even in spontaneous speech, a phrase is uttered at a moment without break, and its syntactic structure is rarely violated. Namely, local syntax is maintained even in ill-formed utterances. Moreover, since a phrase makes a semantic case and is directly mapped into a semantic representation, it will lead to robust understanding.

In the previous section, we proposed a two-pass word spotting strategy which incorporates a simple language model into the non-keyword alternate model as heuristics. It judges the existence of a word w at $t_1 {\sim} t_2$ by evaluating the whole input $f(w)$, not only the target word part $g(w)$. The heuristic language model $h(w)$ constrains the word existence and suppresses false alarms.

$$\begin{aligned} f(w,t_1,t_2) &= h_l(w,1,t_1) + g(w,t_1,t_2) + h_r(w,t_2,T) && (5.3) \\ &= g(w,t_1,t_2) + h(w,t_1,t_2) && (5.4) \end{aligned}$$

Unlike word spotting, it is unrealistic to evaluate all the possible phrases whose number is combinatorial of the vocabulary size. Therefore, some search technique is essential. When we consider sentence understanding, it is significant to spot the N-best candidates correctly. Especially for a large number N, so as to get sufficient candidates, it is hard to obtain the correct N-best candidates with beam search. Thus, we adopt best-first search which outputs the candidates in the order of their scores. This property enables us to choose as many candidates as necessary by adjusting the spotting threshold. This property is also desirable to realize an interaction with an island-driven semantic analyzer.

When we need only the best segmented candidate of a phrase $\mathbf{w} = (w_1, w_2, \ldots, w_l)$, we can apply Viterbi algorithm that computes the score of the optimal path. It is easily extended to N-best algorithm, if we must consider multiple occurrences of the same phrase.

$$f(\mathbf{w}) = \max_{1 \le t_1 < t_2 \le T} f(\mathbf{w}, t_1, t_2) \qquad (5.5)$$

For a partial phrase hypothesis $\mathbf{w}' = (w_1, \ldots, w_n)$, an evaluation function $\hat{f}(\mathbf{w}')$ is defined as an estimate that it completes a phrase.

$$\hat{f}(\mathbf{w}') = g(\mathbf{w}') + \hat{h}(\mathbf{w}') \qquad (5.6)$$

Here, $\hat{h}(\mathbf{w}')$ represents not only the score of the alternate model as in the previous sections but also the prospect for the incompleted part of the phrase.

In other words, $\hat{h}(n)$ combines the estimate score of the incompleted part of the phrase and the score of the rest parts that are not covered by the phrase. The background model is used for constraining the whole utterances. The syntactic and structural analysis is performed on only those parts that are recognizable with the available knowledge. The unrecognizable parts including ill-formed parts are left approximated with the background model only.

If $\hat{h}(\mathbf{w}')$ gives the upper bound of the actual best extension of the phrase, that is, if the linguistic constraint for $\hat{h}(\mathbf{w}')$ is subset of that for the phrase syntax, the search is A*-admissible and guaranteed to find the optimal phrase candidate. In obtaining the N-best candidates as in spotting, the search outputs the hypotheses correctly in the order of their scores. The syllable-concatenation model satisfies this condition, though syllable-bigram does not. The word-concatenation model satisfies it if it covers all of the word entries in the phrase syntax.

Generally, some penalty has to be added to syllable models lest the whole input is matched to the syllables instead of words and phrases. This invalidates the A*-admissibility. However, word-syllable hybrid model still satisfies the admissibility, as it provides the upper bound of scores of possible phrases.

5.2.3 Spotting Algorithm

The spotting algorithm is much the same as word spotting, except that the search proceeds by extending the most plausible phrase hypothesis. To realize best-first search, we use a stack decoder.

The phrase syntax is represented by finite state automaton grammar or LR grammar, by which next possible words can be predicted given the current partial phrase hypothesis. The heuristic language model is incorporated to both left-context and right-context of the hypotheses, as shown in Figure 5.9.

First of all, the heuristic model is applied to the whole part of an input speech. Although the same model is used to approximate the left part and the right part of the spotted phrase, Viterbi scores for the two must be independently computed. Therefore, we apply the model both left-to-right and right-to-left, and stores the respective generated trellises. The third pass is best-first search with a stack decoder that extends plausible

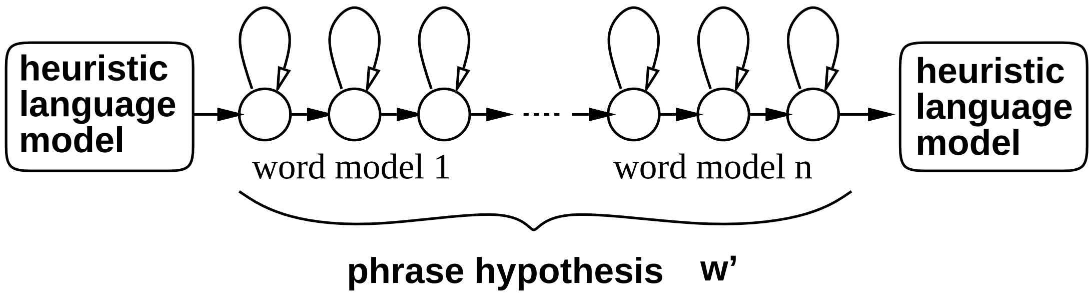

Figure 5.9: Phrase spotting with a heuristic language model

phrase hypotheses. The evaluation function $\hat{f}(\mathbf{w}')$ is obtained by concatenating the phrase part and both heuristics. The algorithm is formulated as follows.

Here, $h_l(w,1,t)$ is the left-context heuristics ending with word w at t, and $h_r(w,t,T)$ is the right-context heuristics starting with word w at time t. The forward score $\alpha(\mathbf{w}',1,t)$ and $\alpha(w,1,t)$ is for a partial phrase hypothesis $\mathbf{w}'$ and for a word w, respectively. We notate a phrase hypothesis of n words as $\mathbf{w}'_n = (w_1,\ldots,w_n)$.

1. The heuristic model is applied left-to-right, and the left-context heuristic trellis $h_l(w_*,1,t_*)$ is computed with Viterbi algorithm.

2. The heuristic model is applied right-to-left, and the right-context heuristic trellis $h_r(w_*,t_*,T)$ is computed with Viterbi algorithm.

3. Words that can appear at the beginning of phrases are predicted. For each w of them, a new hypothesis $\mathbf{w}'_1$ is generated, evaluated with the following Viterbi computation and pushed to the stack.

$$\hat{f}(\mathbf{w}'_1) = \max_{1\le t\le T}\{h_l(w_*,1,t) + h_r(w,t,T)\} \quad (5.7)$$

$$t_1 = \arg\max_{1\le t\le T}\{h_l(w_*,1,t) + h_r(w,t,T)\} \quad (5.8)$$

4. The hypothesis $\mathbf{w}'_n$ (of n words) with the largest score is popped from the stack.

5. If the hypothesis $\mathbf{w}'_n$ completes some phrase that is accepted by the phrase grammar, then output it. If enough candidates are obtained, finish the search.

6. The trellis for the last word of the hypothesis $\mathbf{w}'_n$ is extended.

$$\alpha(\mathbf{w}'_n,1,t) = \begin{cases} \max\limits_{t_0}\{\alpha(\mathbf{w}'_{n-1},1,t_0) + \alpha(w_n,t_0,t)\} & \text{if } n > 1 \\ \max\limits_{t_0}\{h_l(w_*,1,t_0) + \alpha(w_n,t_0,t)\} & \text{if } n = 1 \end{cases} \quad (5.9)$$

7. Words that can extend the hypothesis $\mathbf{w}'_n$ are predicted. For each w of them, a new hypothesis $\mathbf{w}'_{n+1}$ is generated, evaluated with the following Viterbi computation and pushed to the stack. Go to Step 4.

$$\begin{aligned} \hat{f}(\mathbf{w}'_{n+1}) &= \max_{t_{n-1} \leq t \leq T} \{\alpha(\mathbf{w}'_n, 1, t) + h_r(w, t, T)\} && (5.10) \\ t_n &= \arg \max_{t_{n-1} \leq t \leq T} \{\alpha(\mathbf{w}'_n, 1, t) + h_r(w, t, T)\} && (5.11) \end{aligned}$$

The above algorithm assumes that the heuristic model incorporates lexical knowledge, or all the word entries are included in the heuristic model. In this case, the word scores are obtained in the heuristics computation, and the evaluation of the hypotheses (equation (5.7), (5.10)) is computationally economical. Trellis computation for a hypothesis is done only when it is further extended (equation (5.9)).

When we use syllable-level model without lexical knowledge as heuristics, it is necessary to perform word trellis computation at the hypothesis generation. Thus, Step 6 is included in Step 3 and 7. In other words, we have to make computationally expensive evaluation by saving heuristics computation.

As for the terminating condition in Step 5, it is possible to limit the candidates by both the number of the them and their scores. Since the candidates are obtained in the order of their scores with A* search, we can terminate at the desired number of candidates. But it is hard to set the rational number of candidates. Here, we use the score of the candidates as a threshold as in word spotting. The value of the threshold is dynamically set according to the optimal score of the heuristic language model.

5.2.4 Experimental Evaluation

The spotting algorithm is evaluated with speaker-independent continuous speech. The recognition is performed with pair-wise discriminant discrete HMM.

As the phrase grammar, we describe two kinds of grammars. PG1 is normal and fixed, and PG2 allows the ellipses of the functional words, which often occur in conversational Japanese. The word perplexity of PG1 and PG2 is 56.1 and 211.0, respectively.

As the heuristic language model, we use syllable-concatenation model and word-syllable hybrid concatenation model. In both cases, some penalty has to be added to syllable models. Word-pair model turned out not robust in word spotting, thus we do not use it at this time.

Table 5.4: Result of phrase spotting (219 keyword)

language model	grammatical		ill-formed	
(spotting unit) + (heuristics)	A/W	FA/KW/H	A/W	FA/KW/H
phrase (PG1) + word-syllable	67.6	7.23	62.5	6.91
phrase (PG2) + word-syllable	62.2	8.48	56.7	7.97
phrase (PG1) + syllable	47.3	11.45	52.1	8.72
word + word-syllable	61.4	8.66	56.6	7.99
word + syllable	37.6	13.88	42.6	6.77

The spotting accuracy is measured with the detection rate versus the false alarm rate. For consistency with word spotting, we investigate the word detection rate though spotting is done on phrases. The word detection is concerning 219 keywords at the task domain that are used for word spotting. We present the word recognition rate (A/W) and the false alarm rate (FA/KW/H) in the W-best candidates, where W is the number of the words to be spotted in the used samples.

The results for grammatical sentences and ill-formed sentences are shown in Table 5.4. The results of the word spotting are also presented for reference. The detection rates are slightly different from those in the previous section as the occurrence of one word is limited to only once here. The relation between the detection rate (A/W) versus the number of candidates is shown in Figure 5.10 for grammatical utterances and in Figure 5.11 for ill-formed utterances respectively. The number of candidates is measured as a ratio to W.

As the heuristic language model, word-syllable hybrid model brings much better accuracy than syllable-only model. It is confirmed that lexical-level knowledge is significant in the background model for phrase spotting, too.

As for the phrase grammar, grammar PG2 that generates too many phrases gets worse accuracy than normal grammar PG1. Though the decline of the word accuracy is only small, the phrase recognition rate gets almost half. Its phrase recognition rate is even worse than that of grammar PG1 with syllable heuristics.

Compared with simple word spotting, it is clear that the phrase-level syntax is effective. Especially, grammar PG1 improves the detection rate by about 10% with the same heuristics. Actually, the accuracy with word-syllable heuristics is higher than that of word spotting with word-pair constraint. On the other hand, ill-formed grammar PG2 turns out to provide as weak constraint as simple word spotting.

Next, we examine the threshold for spotting. The threshold is set based on the score

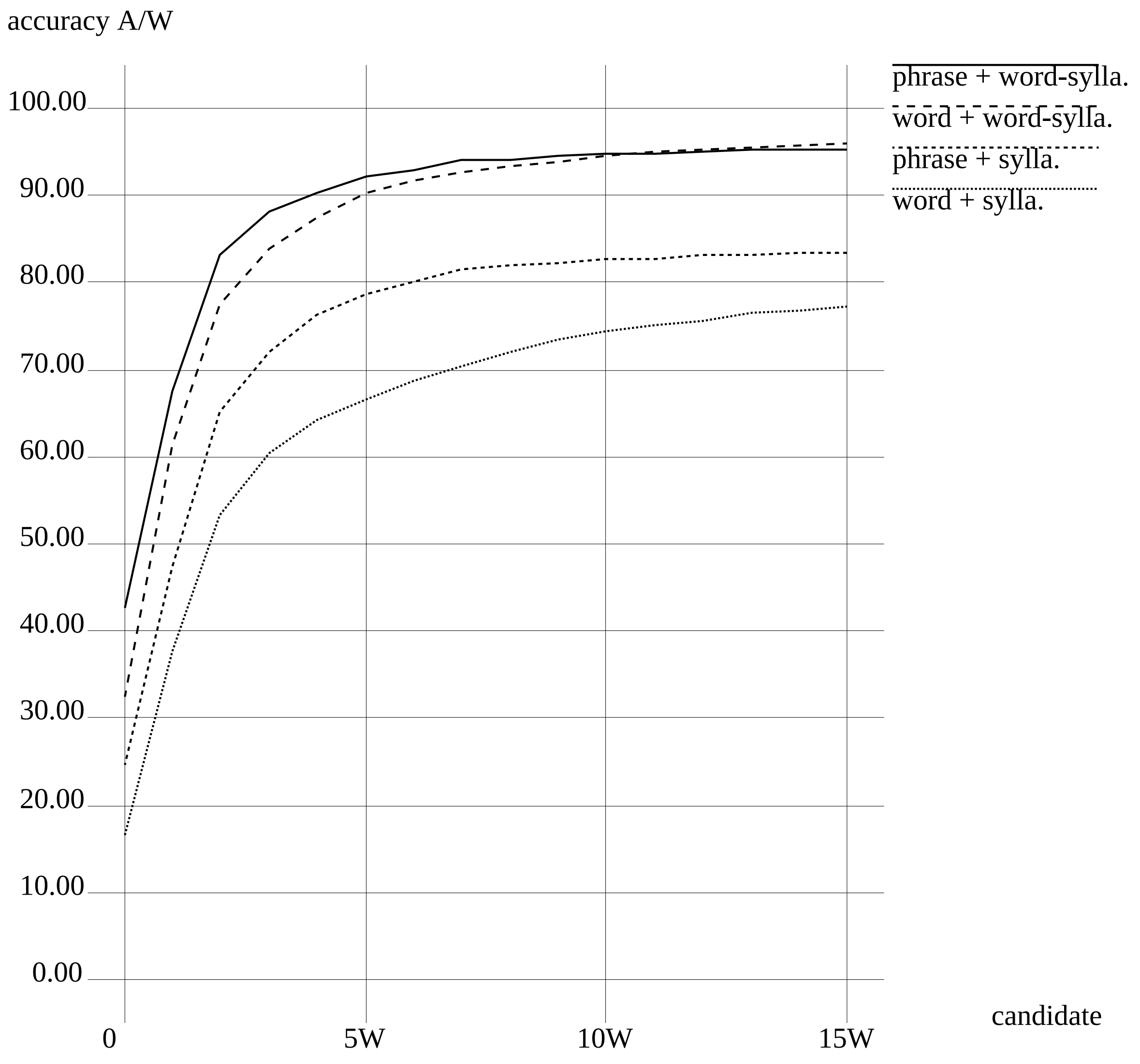

Figure 5.10: Result of phrase spotting (219 keyword; grammatical utterances)

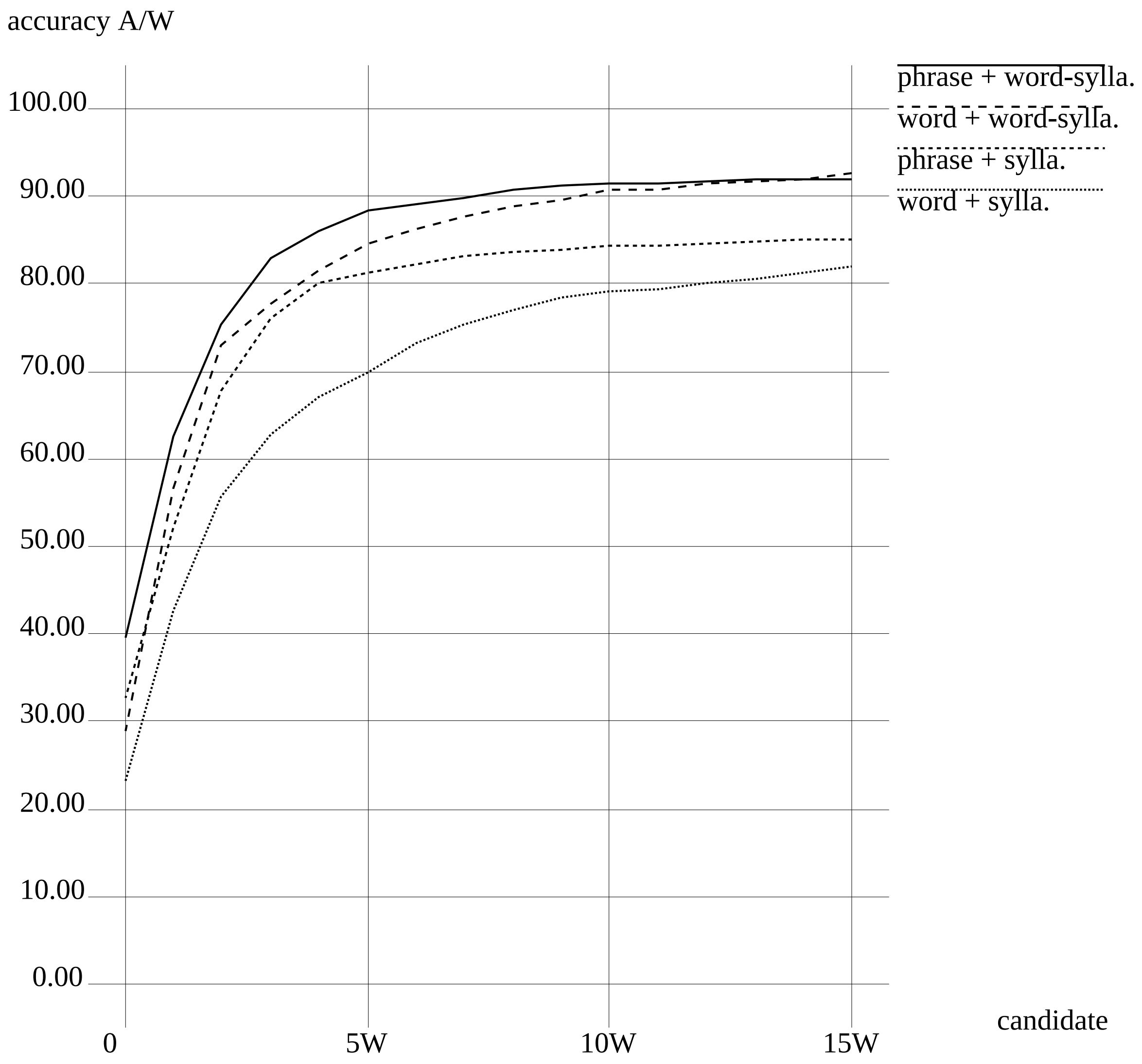

Figure 5.11: Result of phrase spotting (219 keyword; ill-formed utterances)

Table 5.5: Comparison of thresholds for phrase spotting

spotting threshold	grammatical		ill-formed	
	A/P	FA/P	A/P	FA/P
fixed threshold	91.3	25.1	83.4	28.9
length-dependent threshold	94.9	26.3	84.7	19.6

phrase (PG1) + word-syllable

with a heuristic language model P_{max}. In those experiments, we set it to $P_{max} \times a$ (a=1.05) on the logarithm scaled Viterbi score, as in word spotting. However, the threshold must be dependent on the length of the input. In the longer input utterances, the part of the heuristic score gets larger and the ratio of the hypothesis score to P_{max} gets smaller. Therefore, we make the weight a variable on the length T as follows.

$$a = c/T + 1.0 \qquad\qquad c: \text{constant}$$

Here, we set c=16.0 where T is the number of 10-ms frames.

The comparison of the thresholds are shown in Table 5.5. Here, we present the phrase recognition rate (A/P) and the false alarm rate (FA/P) in all spotted candidates with respect to the number of the correct phrases P, not words. The overall quality of the phrase candidates is improved with the threshold dependent on the input length. Especially, the number of false alarms in ill-formed utterances is significantly reduced, with much the same detection rate. It means the reduction of the search space of the following sentence understanding process.

5.2.5 Conclusion

We have proposed a phrase spotting strategy for robust recognition of spontaneous speech. It uses a word or syllable model as heuristics which is robust against ill-formed utterances. As the heuristic constraint is subset of the phrase syntax, A* search is realized which outputs the spotted candidates in the order of their scores. This best-first property is significant in getting appropriate phrase candidates without generating vain hypotheses.

Experimental results show that the phrase-level syntax improves the word detection rate significantly compared with simple word spotting.

5.3 Search Strategy for Spotting-based Understanding

5.3.1 Introduction

For robust spoken language understanding, we have studied word and phrase spotting algorithms. In order to realize actual understanding, we need a mechanism to combine the results of the spotting and construct sentence hypotheses.

In this section, we explore the search strategy for sentence parsing based on the results of the spotting. By retaining the intermediate results of the spotting, it is possible to realize A* search that obtains the optimal sentence candidate with correct Viterbi score.

The search algorithms are studied with respect to the direction of the search. Then, knowledge sources to drive the search are also examined.

5.3.2 Heuristic Search based on Phrase Spotting

In the conventional spotting-based recognition, deterministic segmentation is performed on the spotted candidates[71],[78]. They are represented with their starting points, end points and scores. Thus, the set of candidates is called a word/phrase lattice that is ordered on the time dimension.

This compact representation is advantageous to save computation of the following processing that combines the candidates. However, it gets impossible to compute correct Viterbi score of the combination of the candidates. Viterbi computation performs maximization over possible concatenations of the segments to get the globally optimal score. The omission of the optimization may be critical for overall recognition, though it may be overlooked.

Therefore, we keep trace of the intermediate result of the spotting that contains sufficient information to perform Viterbi computation. Concretely, we retain the trellises of the initial or the final state of the candidates. In combining the spotted candidates, the trellises are merged over the time dimension to get the maximum score. This manipulation is expected to be equivalent to the continuous speech parsing that computes the globally optimal Viterbi score.

Moreover, it can be implemented with A*-admissible search in the same way as the phrase spotting, which uses the same heuristics consequently. Together with phrase spotting, overall recognition is formulated as a progressive search, which first applies heuristic

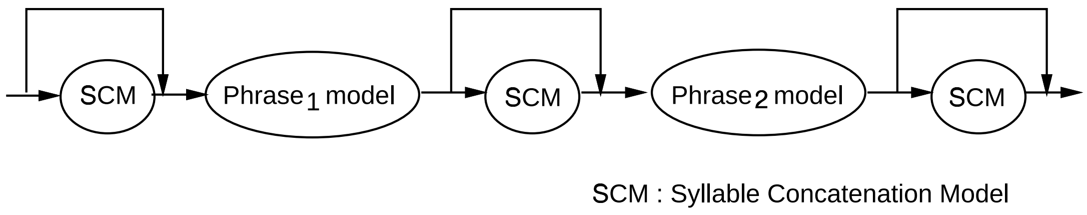

Figure 5.12: Sentence hypothesis model based on phrase spotting

language level, then the intra-phrase syntax, and finally the inter-phrase constraint.

Compared with continuous speech parsing, the components of hypotheses are limited to the spotted candidates. This might be also the loss of information. However, continuous speech parsing often causes hypothesis explosion for utterances with unknown words or out-of-syntax expressions. Thus, spotting-based approach that skips them are advantageous.

5.3.3 Search Algorithms

A sentence hypothesis consists of several spotted phrases and unrecognized parts such as fillers. The skipped parts are approximated with syllable-concatenation model. Thus, the sentence hypothesis is modeled as shown in Figure 5.12. Here, some penalty for skipping has to be added to the syllable models lest the whole input is matched to the syllables instead of words and phrases, as in the phrase spotting stage.

As heuristics for the search, we use word-syllable hybrid model that is also used for the phrase spotting. It is admissible since it provides the upper bound of the score for both the phrase model and the syllable models with the same penalty.

The search algorithm is fundamentally A* search that extends the currently best hypothesis. It is formulated as follows.

1. For each phrase candidate, an initial sentence hypothesis is generated and pushed to the stack with its evaluation score.

2. The hypothesis with the largest score n is popped from the stack.

3. If the hypothesis n has an end flag, then output it. Finish the search.

4. If the hypothesis n complete a sentence, Viterbi score of the sentence model is computed and pushed to the stack with an end flag.

5. Phrases that can extend the hypothesis n are predicted. For each of them, a new sentence hypothesis is generated and pushed to the stack with its evaluation score. Go to Step 2.

Here, a hypothesis completes a sentence if it satisfies the semantic constraint for a sentence, for example, it is accepted by a grammar. Notice that the coverage of the input speech is not counted as the terminating condition. In the search stage, the score normalization is done with the heuristic score for the uncovered part that is embedded in the phrase score. However, its evaluation score does not necessarily correspond to the model shown in Figure 5.12 that covers the whole input without heuristics. Therefore, the strict Viterbi computation is performed for a complete hypothesis in Step 4. If it is still best among the hypotheses with admissible heuristics, then it is certainly the optimal solution to be output. If sufficient number of hypotheses are spotted, the algorithm combined with the phrase spotting gives the globally optimal solution.

Then, we describe concrete algorithms for hypothesis evaluation with respect to the direction of search: right-to-left and island-driven.

Right-to-Left Search

First, we consider simple one-directional search that proceeds on the time dimension. The direction of the search must be reverse of that of the phrase spotting for evaluation described below. Here we perform right-to-left search after left-to-right phrase spotting. Its search space is a tree of the spotted phrases, thus the evaluation and management of the hypotheses get simple.

A partial sentence hypothesis is modeled as shown in Figure 5.13 and evaluated with the heuristic model. The score for this model is computationally economical.

Suppose that the currently poped hypothesis is n, and a new hypothesis is generated by concatenating a phrase A. The evaluation function for the new hypothesis is computed by the following trellis computation, that is a concatenation of the backward trellis of the hypothesis including the syllable model $\beta_n(t)$, and the forward trellis of the spotted phrase A together with heuristic language model $\alpha_A(t)$.

$$\hat{f}(n+A) = \max_t\{\alpha_A(t) + \beta_n(t)\} \tag{5.12}$$

Here, the trellis for the phrase A is not generated and the result of the phrase spotting

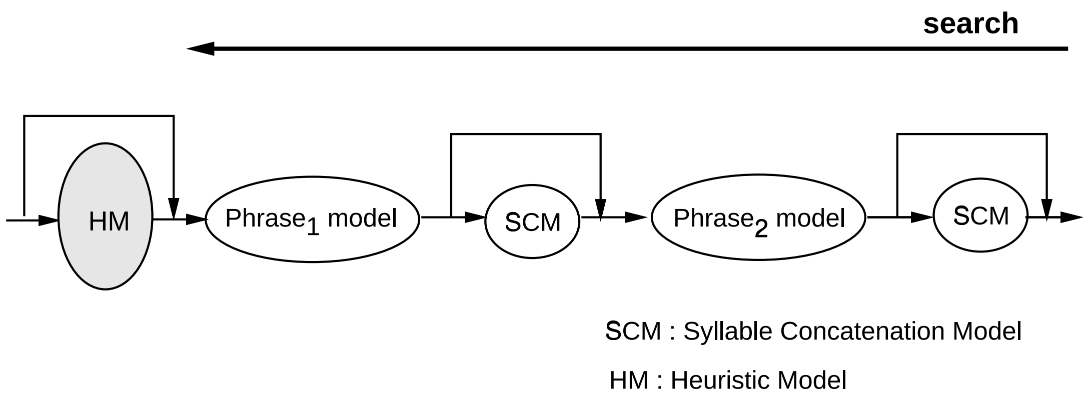

Figure 5.13: Right-to-left search based on phrase spotting

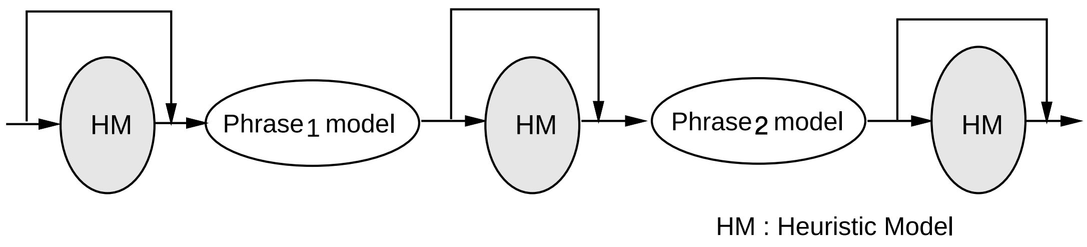

Figure 5.14: Island-driven search based on phrase spotting

is utilized instead in Step 5. The trellis for the hypothesis is extended as $\beta_n(t)$ only when it is popped from the stack.

Though the search mechanism is simple, right-to-left search does not necessarily proceed on the good paths that will overcome the ill-formedness.

Island-Driven Search

Next, we consider island-driven search that starts with the most plausible part and proceeds by concatenating better phrases in order. It proceeds rather on the score dimension and is independent from the time dimension. A new phrase can be inserted between the phrases of a hypothesis. This property implies that the same hypothesis can be generated from different ones. For example, both hypothesis $\{A, B\}$ and hypothesis $\{A, C\}$ can generate hypothesis $\{A, B, C\}$. Namely, the search space is a graph of the spotted phrases, not a tree. Moreover, the search tends to generate so many hypotheses. Thus, the mechanism to manage hypotheses gets complex.

A partial sentence hypothesis is modeled as shown in Figure 5.14. Here, the heuristic model represents both the unsearched phrases and the skipped parts.

In this case, it is impossible to get Viterbi score of this model with the currently best hypothesis and a spotted phrase. Therefore, we introduce an approximate evaluation function. Suppose that the currently poped hypothesis is $n = \{A, C\}$, and a new hypothesis is generated by concatenating a phrase B between the phrase A and the phrase C. For every time-frame t of the trellis $\alpha_W(t)$ of every spotted phrase W, its starting time-frame $st_W(t)$ is back-traced and stored at the spotting stage. Then, the score for the phrase B is approximated with the following.

$$\begin{aligned}
\hat{f}_p(B) &= \max_{end_A < st_B(t),\ t < start_C} \alpha_B(t) & (5.13)\\
end_B &= \arg \max_{end_A < st_B(t),\ t < start_C} \alpha_B(t) & (5.14)\\
start_B &= st_B(end_B) & (5.15)
\end{aligned}$$

The evaluation function for the new hypothesis $\{A, B, C\}$ is computed as an offset from the optimal heuristic score for the whole input $\hat{h}_0$.

$$\begin{aligned}
\hat{f}(ABC) &= \hat{h}_0 - (\hat{h}_0 - \hat{f}_p(A)) - (\hat{h}_0 - \hat{f}_p(C)) - (\hat{h}_0 - \hat{f}_p(B)) & (5.16)\\
&= \hat{f}(AC) - (\hat{h}_0 - \hat{f}_p(B)) & (5.17)
\end{aligned}$$

It is initialized as below.

$$\hat{f}(null) = \hat{f}_p(null) = \hat{h}_0, \quad end_{null} = 0, \quad start_{null} = T$$

This algorithm is based on the same concept as the shortfall method[55]. Every time a new phrase is added, its offset from the optimal heuristic score is subtracted. Strictly speaking, the algorithm is not admissible due to the Viterbi segmentation. But it gives a reasonable evaluation.

Here we perform Viterbi segmentation at every step of the hypothesis extension, which deviates the correct score. At the very last step when the hypothesis complete a sentence, the correct score is computed with the model over the whole input at Step 4. The actual trellis computation is done only in this step. Thus, the overall amount of computation depends on how many sentence candidates are completed until one of them is output, and how precise the above evaluation function is.

5.3.4 Knowledge Sources

Next we examine the knowledge sources that drive the search. They are required to predict possible phrase candidates to extend a current partial sentence hypothesis and to accept a completed sentence.

Inter-Phrase LR grammar

An inter-phrase grammar describes the possible sentence patterns with rewriting rules whose terminals are phrases. It is more abstract than grammars for continuous speech recognition, and represents semantics and pragmatics rather than syntax.

The search algorithm is much the same way as in normal LR parsing, except that the possible terminal symbols are limited to the spotted phrases. But it is very troublesome to implement island-driven search that needs bi-directional prediction. One-directional search such as right-to-left search is regarded as a simple robust parsing by skipping unrecognizable parts.

Semantic Network

A semantic network that is introduced in Section 4.3 represents the relationship between concepts and words. Unlike an LR grammar, it basically does not constrain the order of surface words or phrases. Therefore, it realizes genuinely semantic-driven understanding. Island-driven search is possible as well as left-to-right search.

In Section 4.3, we discussed cooperation of the speech recognizer and the semantic analyzer, where sentence hypotheses are generated by the speech recognizer and interactively verified by the semantic network. Here, the semantic network takes the initiative and constructs sentence hypotheses.

The semantic network must have ability to predict the following hypotheses given a current partial sentence. In sentence understanding with marker passing, the semantic markers are passed upward from word nodes to concept nodes. In prediction, the markers are then passed downward from the currently activated concept nodes to word nodes, and consistent word nodes are picked up. There are two kinds of prediction. One is the proper set that must appear to complete the sentence hypothesis. The other is the permissible set that is not essential but may appear. In island-driven search, the proper set of phrases can be extended with priority, while one-directional search has to consider all the possible phrases.

5.3.5 Experimental Evaluation

We have evaluated above search algorithms and knowledge representations with continuous speech understanding. The experiments are done using the results of the phrase

Table 5.6: Comparison of several approaches for sentence understanding

	grammatical			ill-formed		
approach	word	sent.	semantic	word	sent.	semantic
phrase spot. + grammar (w/o syllable)	81.5	60.5	66.8	72.7	17.5	43.0
phrase spot. + grammar (with syllable)	81.1	49.0	59.8	75.0	24.0	45.5
word-pair	82.9	50.3	53.3	72.3	10.0	23.5
word-pair (+syllable)	78.3	40.3	43.5	67.4	11.5	20.5
sentence parser	86.2	65.0	68.3	59.9	13.5	36.5
sentence parser (+syllable)	82.3	56.0	60.5	61.3	22.0	37.0
word spot.			44.0			27.5

spotting with word-syllable hybrid model as heuristics. Its performance was best at the experiments and listed in Table 5.5. Here, the number of phrases per utterance is limited to 150 at maximum.

At first, we implemented simple right-to-left search with an inter-phrase grammar as the knowledge source. The stack size of this A* search is set to 100 and the number of generated hypotheses is limited to 500. The inter-phrase grammar was described based on LR grammar G1 for continuous speech recognition. The obtained sentence candidates are processed by the semantic network module to get a semantic representation.

This spotting-based approach is compared with word-pair and sentence-level LR parser and word spotting, using the same semantic module. Here, we use LR grammar G1 that describes the syntax of the whole sentences. Its word perplexity is 17.2 and it can cope with silent pauses and some sorts of filled pauses. A word-pair grammar is also derived from the LR grammar. Its word perplexity is 32.7. Word spotting is based on word-pair heuristics.

Syllable-concatenation model is incorporated to the sentence-level LR parser and the word-pair to model fillers between phrases. But we also performed recognition without the syllable model to see its effect. In this case, only silent pauses can be filled between phrases.

The word recognition rate (word), the sentence recognition rate (sent.), and the semantic accuracy (semantic) are shown in Table 5.6. The semantic accuracy is the ratio of the samples whose semantic representations are correctly obtained.

For the grammatical utterances, the sentence-level LR parser that uses the most constraining syntax and performs global optimization outperforms the others. The phrase spotting-based approach decreases the recognition rate by about 5%, due to limiting the intermediate candidates. Use of the syllable model lowers the accuracy much, because it substantially increases the perplexity of the task. Moreover, these samples are covered with the registered words and syntax. The word-pair grammar, which provides only local constraint, gets the worst sentence rate though its word accuracy is comparable.

For the ill-formed utterances, however, the phrase spotting-based approach gets the best accuracy. The sentence-level LR parser cannot cope with some of the ill-formedness at all and the search often fails to output any candidates. Thus, it gets the worst word rate. Use of the syllable model has little effect. It actually works as a filler but matches with non-fillers on the other hand. The simple word-pair constraint gets a good word recognition rate, but it does not lead to robust sentence-level understanding.

The word spotting-based approach that also uses word-pair constraint only and performs deterministic segmentation of words cannot achieve good accuracy.

In the experiments, the inter-phrase grammar combined with the intra-phrase grammar gives much the same constraint as that by the sentence-level LR grammar. As the search strategy, however, a different approach is appropriate according to the utterance characteristics. For the grammatical utterances, continuous sentence parsing with global optimization is effective. For the ill-formed utterances, the spotting-based approach is robust, as it does not cause hypothesis explosion

Knowledge Source

Next, we examine the knowledge sources to combine the spotted phrases. In addition to the inter-phrase grammar, we also implemented the search driven by a semantic network. We also performed recognition without any knowledge sources, where any phrases can be combined. This is possible because the number of the phrases is so limited.

They are realized with right-to-left search. The experiments are done both with and without the syllable model. The results are shown in Table 5.7.

The inter-phrase grammar is more effective than the semantic network. The constraint provided by the semantic network is weak, especially in one-directional search. Actually, it gets much the same accuracy as that without any knowledge sources.

Table 5.7: Comparison of knowledge sources for phrase spotting-based approach

right-to-left search

	grammatical			ill-formed		
knowledge source	word	sent.	semantic	word	sent.	semantic
inter-phrase grammar	81.5	60.5	66.8	72.7	17.5	43.0
	81.1	49.0	59.8	75.0	24.0	45.5
semantic network	77.8	46.8	54.3	61.1	16.0	30.5
	74.1	40.0	48.0	67.3	19.0	31.5
none	74.2	42.0	53.3	61.5	14.5	28.0
	73.0	34.3	47.0	63.3	15.5	27.5

upper row: without syllable model
lower row: with syllable model

Table 5.8: Comparison of search direction for phrase spotting-based approach

semantic network with syllable model

	grammatical			ill-formed		
direction of search	word	sent.	semantic	word	sent.	semantic
right-to-left	74.1	40.0	48.0	67.3	19.0	31.5
island-driven	56.9	38.5	44.0	43.0	16.0	27.0

Direction of Search

Then, we realize island-driven search that will be suitable for the semantic network. The comparison with right-to-left search is done in Table 5.8. The syllable model is used in the both cases.

Island-driven search fails to output any candidates more often than right-to-left search. It generates sentence hypotheses in both directions, and the number of the hypotheses explodes. Therefore, the word recognition rate gets worse, though the sentence accuracy is much the same. The prediction mechanism of the semantic network has much room to be improved.

From the computational aspect, the number of extended partial sentence hypotheses in right-to-left search is 122, while island-driven search makes 80 complete sentence hypotheses for the ill-formed utterances. Since a sentence consists of 3 phrases on the average, the trellis computation of island-driven search is about twice of that of right-to-left search.

5.3.6 Conclusion

We have discussed the search strategy to combine the spotted phrases to get the most plausible sentence hypothesis. The spotted phrases are represented with its trellis scores so as to get correct Viterbi score of the combined sentence hypothesis. This mechanism is essential in order to get performance comparable to continuous speech recognition. It also realize A* search with the same heuristics as in the phrase spotting.

The search algorithms are formulated both one-directional and island-driven. One-directional search is realized in the same manner as the phrase spotting itself. For island-driven search, an efficient evaluation function is introduced. As for the knowledge sources to drive the search, an inter-phrase grammar and a semantic network are used. Use of semantic network realizes a thorough semantic-driven search.

Experimental results of speech understanding shows that the phrase spotting-based approach is more effective for the ill-formed utterances than continuous speech recognition with a sentence-level grammar or a word-pair grammar. It extracts only recognizable parts and directly leads to robust sentence understanding.

Chapter 6

Conclusion

We have studied various issues for speech recognition and understanding.

As for acoustic modeling, we have proposed an approach that realizes both discriminative ability and robustness. Pair-wise discriminant analyses are embedded as the front-end of HMM to fully separate competing classes. It is implemented both as discrete HMM and continuous HMM, and demonstrated to outperform the conventional HMMs. We have also shown the significance of stochastic scoring of the phone model for continuous speech recognition.

The proposed model needs much computation. However, it is getting possible to process in real-time with a recent powerful workstation. The most important property is that it is a combination of simple classifiers. Though the total number of parameters is large, the same training samples are used to estimate different discriminant functions. Therefore, the recognizer can be constructed with quite a small number of samples.

With respect to the search strategy that integrates the acoustic model and the language model, we have addressed admissible and efficient algorithms for both continuous speech parsing and word/phrase spotting. They incorporate low-level language models as heuristics, which guides the following search pass. With lexical-level heuristics such as word-pair constraint, the progressive search finds the optimal candidates quite efficiently in the order of their scores.

The algorithms are realized at the expense of virtual real-timeness. However, the recent computing power enables it to work in quick turn-around time. The intensive computation is in the first pass that can be processed in real-time. Especially, continuous speech parsing algorithm can obtain the results with the simple constraint at the first pass, for the utterances that are clear both acoustically and linguistically. Actually, about 80%

of the optimal candidates with sentence LR parsing are the best sequences at the first pass. In a sense, the following search can be regarded as backtracking the input by applying higher linguistic knowledge.

In this formulation of the search strategy, multiple knowledge sources are integrated. They are not used uniformly, but for different appropriate purposes, maintaining their modularity. In principle, we use statistical knowledge sources such as syllable or word bigram for decoding input phone or word sequences. The intermediate results are used as heuristics to guide the following process of structuring and understanding the utterances. At this process, we have proposed an interaction between the syntactic knowledge sources such as an LR grammar and the conceptual knowledge sources such as a semantic network. Although the modularity is maintained, the progressive and cooperative search does not allow any loss of information, thus realizes a genuine integration. Concretely, we have pointed out the significance of integrating knowledge sources and the harmfulness of a deterministic intermediate representation such as a segmented word lattice.

We have discussed the two approaches for speech understanding: parsing and spotting. The parsing approach is adequate for processing grammatical sentences with high accuracy and efficiency, while the spotting approach is for dealing with ill-formed utterances. An optimal approach should be taken, according to the expected ratio of grammatical utterances and ill-formed ones. When we can generally predict and cover the patterns of utterances as in a rather restricted task, use of a sentence-level grammar will be the best. If we allow variety of utterances that cannot be expected beforehand, the spotting strategy is desirable. The conditions are significantly affected by the task design and the interface of the application.

In general, the strategy to switch them is desirable[79],[80]. Since not a few part of the utterances are well-formed, parsing should be performed at first. And if it fails, then spotting should be applied. Since the basis of the two approaches are common in our methods, namely the common heuristics can be shared by the both, it is easy to switch them without overhead computation. The strategy is illustrated in Figure 6.1. In our experiments, when we take this hybrid strategy, we can get the semantic accuracy of 40.5% for ill-formed utterances, while keeping 69.0% for grammatical ones.

Thus, we have realized a speaker-independent continuous speech recognition system. It is integrated with a natural language processor and a task manager into a spoken dialogue system.

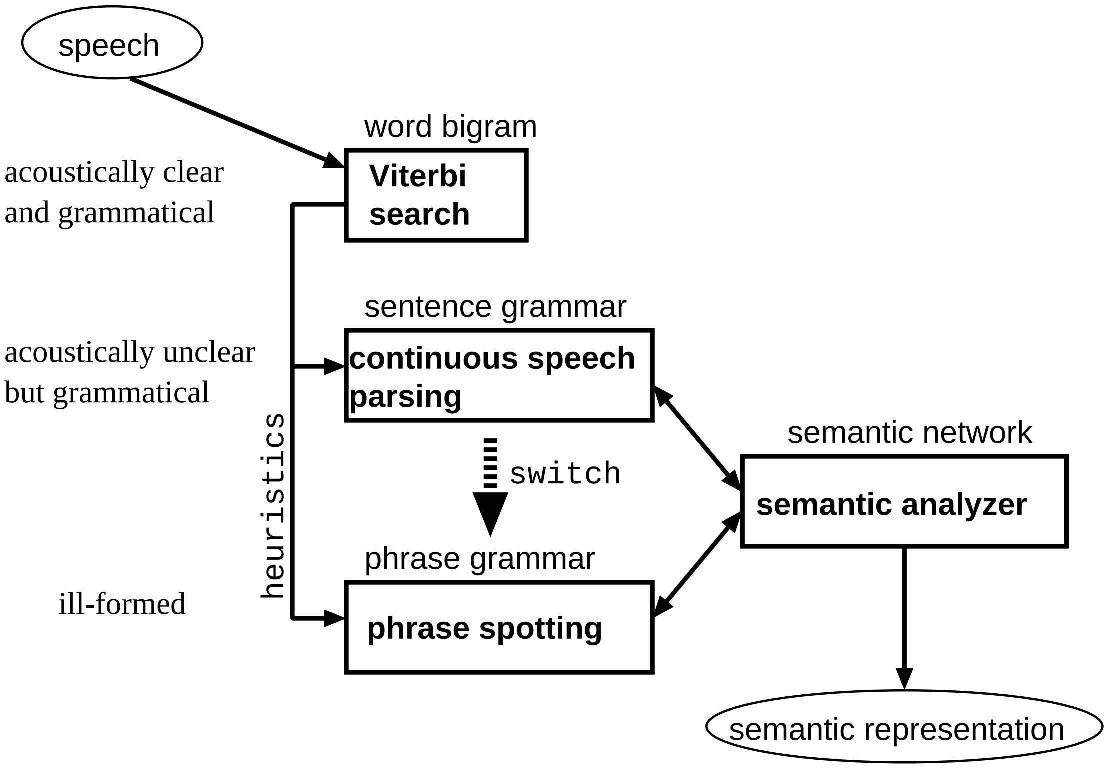

Figure 6.1: Search strategy that switches from parsing to spotting

It is needless to point out the necessity of the further improvement of its baseline performance. In addition, the robustness is the vital factor. Though our study is oriented for it, the robustness is an open problem in nature, and it is endless to evaluate it.

Especially, in a spoken dialogue system, it is hard to predict the knowledge and the intentions of the users, thus their utterances themselves. The dialogue modeling and control is significant in order to lead them to correspond with the knowledge and the capability of the system. The system we have built will be useful for the further analysis and improvement.

Automatic learning and adaptation of the system is also significant. The speaker adaptation technique of the speech recognizer has been studied and demonstrated. However, construction of the language models, especially concerning the task knowledge, needs much human effort and time. Automatic adaptation and semi-automatic initial construction of the module is desirable.

Acknowledgements

The study has been accomplished in Kyoto University since I entered graduate school. I would like to express my sincere gratitude to Professor Shuji Doshita, my supervisor. Without his enlightening guidance and warmful support, I could not have completed the work.

I am grateful to Professor Katsuo Ikeda and Professor Makoto Nagao for their invaluable comments to the thesis.

I also greatly appreciate Professor Shigeyoshi Kitazawa of Shizuoka University and Professor Toyoaki Nishida of Nara Institute of Science and Technology (NAIST) for their fruitful comments and encouragement through the research.

I would like to thank Professor Toshiyuki Sakai and Associate Professor Michihiko Minoh, who introduced me to the research activities.

I owe a great deal to the members of Professor Doshita's Laboratory. I had many suggestions from Mr. Hiroaki Kojima (currently at ETL), Dr. Atsushi Yamada (currently at NAIST) and Mr. Masahiro Araki. I also discussed quite a lot with Ms. Pascale Fung (currently at Columbia University), who also reviewed a part of the thesis. I am also indebted to Mr. Yoichi Mizutani, Mr. Toru Ogawa, Mr. Shinji Matsumoto, Mr. Toshihiko Munetsugu, Mr. Norihide Kitaoka, Mr. Nobuo Nukaga, and Mr. Kiyokazu Miki for their cooperation in the work.

Finally, I wish to thank my family and friends for their support.

Bibliography

[1] S.Doshita. *Studies on the Analysis and Recognition of Japanese Speech Sounds.* PhD thesis, Kyoto University, 1965.

[2] K.H.Davis, R.Biddulph, and S.Balashek. Automatic recognition of spoken digits. *J. Acoust. Soc. Am.*, 24(6):637–642, 1952.

[3] F.Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoust., Speech & Signal Process.*, 23(1):67–72, 1975.

[4] H.Sakoe and S.Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust., Speech & Signal Process.*, 26(1):43–49, 1978.

[5] M.Kohda, R.Nakatsu, and K.Shikano. Speech recognition in the question answering system operated by conversational speech. *Proc. of IEEE-ICASSP*, 1976.

[6] V.R.Lesser, R.D.Fennell, :.D.Erman, and R.D.Reddy. The Hearsay-II speech understanding system. *IEEE Trans. Acoust., Speech & Signal Process.*, 23(1):11–24, 1975.

[7] B.T.Lowerre. *The HARPY speech understanding system.* PhD thesis, Carnegie-Mellon University, 1976.

[8] S.Nakagawa. *A Machine Understanding System for Spoken Japanese Sentences.* PhD thesis, Kyoto University, 1976.

[9] F.Jelinek. Continuous speech recognition by statistical methods. In *Proc.IEEE*, volume 64, pages 532–556, 1976.

[10] L.R.Rabiner, S.E.Levinson, and M.M.Sondhi. On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition. *Bell Syst. Tech. J.*, 62(4):1075–1105, 1983.

[11] K.F.Lee and H.W.Hon. Large-vocabulary speaker-independent continuous speech recognition using HMM. In *Proc. of IEEE-ICASSP*, pages 123–126, 1988.

[12] F.Jelinek and R.L.Mercer. Interpolated estimation of Markov source parameters from sparse data. In E.S.Gelsema and L.N.Kanal, editors, *Pattern Recognition in Practice.* North-Holland, Amsterdam, 1980.

[13] K.F.Lee. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Nerwell, 1989.

[14] D.E.Rumelhart and J.L.McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, 1986.

[15] N.J.Nilsson. *Learning Machines.* McGraw-Hill, New York, 1965.

[16] R.Schwartz and S.Austin. A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses. In *Proc. of IEEE-ICASSP*, pages 701–704, 1991.

[17] S.Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. Acoust., Speech & Signal Process.*, 34(1):52–59, 1986.

[18] L.R.Bahl, P.F.Brown, P.V.de Souza, and R.L.Mercer. Speech recognition with continuous-parameter hidden Markov models. In *Proc. of IEEE-ICASSP*, pages 40–43, 1988.

[19] R.Schwartz, Y.Chow, O.Kimball, S.Roucos, M.Krasner, and J.Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *Proc. of IEEE-ICASSP*, pages 1205–1208, 1985.

[20] J.Takami and S.Sagayama. A successive state splitting algorithm for efficient allophone modeling. In *Proc. of IEEE-ICASSP*, volume 1, pages 573–576, 1992.

[21] G.R.Doddington. Phonetically sensitive discriminants for improved speech recognition. In *Proc. of IEEE-ICASSP*, pages 556–559, 1989.

[22] L.R.Bahl, P.F.Brown, P.V.de Souza, and R.L.Mercer. Maximum mutual information estimation of hidden Markov parameters for speech recognition. In *Proc. of IEEE-ICASSP*, pages 49–52, 1986.

[23] E.McDermott and S.Katagiri. Prototype-based discriminative training for various speech units. In *Proc. of IEEE-ICASSP*, volume 1, pages 417–420, 1992.

[24] L.R.Bahl, P.F.Brown, P.V.de Souza, and R.L.Mercer. A new algorithm for the estimation of hidden Markov model parameters. In *Proc. of IEEE-ICASSP*, pages 493–496, 1988.

[25] J.Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, 1970.

[26] A.V.Aho and J.D.Ullman. *The Theory of Parsing, Translation, and Compiling, Vol.1: Parsing.* Prentice-Hall, 1972.

[27] M.Tomita. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, 1985.

[28] L.R.Bahl, F.Jelinek, and R.Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5(2):179–190, 1983.

[29] R.Schwartz and Y.L.Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proc. of IEEE-ICASSP*, pages 81–84, 1990.

[30] Acoustical Society of Japan. *Continuous Speech Corpus for Research*, 1993.

[31] S.K.Katz. Estimation from sparse data for the language model for a speech recognition. *IEEE Trans. Acoust., Speech & Signal Process.*, 35(3):400–401, 1987.

[32] N.Abramson. *Information Theory and Coding.* McGraw-Hill, New York, 1963.

[33] E.Charniak. *Statistical Language Learning.* MIT Press, Cambridge, 1993.

[34] K.Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, 1972.

[35] *BMDP Statistical Software Manual*, 1988.

[36] S.Doshita, S.Kitazawa, M.Ishikawa, H.Kojima, and Y.Nishimura. Discrimination on Japanese stop consonants using pair-wise discrimination method. *STUDIA PHONOLOGICA*, 20:71–79, 1986.

[37] Lachenbruch P.A. *Discriminant Analysis.* Hafner Press, New York, 1975.

[38] Y.Linde, A.Buzo, and R.M.Gray. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, COM-28(1):84–95, 1980.

[39] H.Iwamida, S.Katagiri, E.McDermott, and Y.Tohkura. A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proc. of IEEE-ICASSP*, pages 489–492, 1990.

[40] W.Ma and D.V.Compernolle. TDNN labeling for a HMM recognizer. In *Proc. of IEEE-ICASSP*, pages 421–423, 1990.

[41] H.P.Tseng, M.J.Sabin, and E.A.Lee. Fuzzy vector quantization applied to hidden Markov modeling. In *Proc. of IEEE-ICASSP*, pages 641–644, 1987.

[42] X.D.Huang an M.A.Jack. Unified techniques for vector quantization and hidden Markov models using semi-continuous models. In *Proc. of IEEE-ICASSP*, pages 639–642, 1989.

[43] J.R.Bellegarda and D.Nahamoo. Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In *Proc. of IEEE-ICASSP*, pages 13–16, 1989.

[44] P.Ramesh, S.Katagiri, and C.H.Lee. A new connected word recognition algorithm based on HMM/LVQ segmentation and LVQ classification. In *Proc. of IEEE-ICASSP*, pages 113–116, 1991.

[45] G.Yu, W.Russell, R.Schwartz, and J.Makhoul. Discriminant analysis and supervised vector quantization for continuous speech recognition. In *Proc. of IEEE-ICASSP*, pages 685–688, 1990.

[46] L.A.Liporace. Maximum likelihood estimation for multivariate observations of Markov process. *IEEE Trans. Inf. Theory*, IT-28(5):729–734, 1982.

[47] N.Morgan and H.Bourlard. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *Proc. of IEEE-ICASSP*, pages 413–416, 1990.

[48] M.Franzini, K.F.Lee, and A.Waibel. Connectionist Viterbi training: A new hybrid method for continuous speech recognition. In *Proc. of IEEE-ICASSP*, pages 425–428, 1990.

[49] K.Y.Su and C.H.Lee. Robustness and discrimination oriented speech recognition using weighted HMM and subspace projection approaches. In *Proc. of IEEE-ICASSP*, pages 541–544, 1991.

[50] P.C.Woodland and D.R.Cole. Optimizing hidden Markov models using discriminative output distributions. In *Proc. of IEEE-ICASSP*, pages 545–548, 1991.

[51] T.Kawahara, T.Ogawa, S.Kitazawa, and S.Doshita. Phoneme recognition by combining Bayesian linear discriminations of selected pairs of classes. In *Proc. Int'l Conf. on Spoken Language Processing*, 7.8, 1990.

[52] T.Kawahara, S.Matsumoto, and S.Doshita. A*-admissible context-free parsing on HMM trellis for speech understanding. In *Proc. Pacific Rim Int'l Conf. on Artificial Intelligence*, volume 2, pages 1203–1208, 1992.

[53] K.Nagao, K.Hasida, and T.Miyata. Understanding spoken natural language with omni-directional information flow. In *Proc. of IJICAI*, pages 1268–1274, 1993.

[54] N.J.Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York, 1971.

[55] W.A.Woods. Optimal search strategies for speech understanding control. *Artificial Intelligence*, 18:295–326, 1982.

[56] D.B.Paul. Algorithms for an optimal A* search and linearizing the search in the stack decoder. In *Proc. of IEEE-ICASSP*, pages 693–696, 1991.

[57] D.B.Paul. An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *Proc. of IEEE-ICASSP*, volume 1, pages 25–28, 1992.

[58] 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. of IEEE-ICASSP*, pages 705–708, 1991.

[59] S.Austin, R.Schwartz, and P.Placeway. The forward-backward search algorithm. In *Proc. of IEEE-ICASSP*, pages 697–700, 1991.

[60] T.Yamaoka and H.Iida. A method to predict the next utterance using a four-layered plan recognition model. In *Proc. of ECAI*, pages 726–731, 1990.

[61] T.Yamamoto, Y.Ohta, Y.Yamashita, O.Kakusho, and R.Mizoguchi. Mascots: Dialogue management system for speech understanding system. *IEICE Trans.*, E74(7):1881–1888, 1991.

[62] A.G.Hauptmann, S.R.Young, and W.H.Ward. Using dialog-level knowledge sources to improve speech recognition. In *Proc. of AAAI*, pages 729–733, 1988.

[63] M.Nagata. Using pragmatics to rule out recognition errors in cooperative task-oriented dialogues. In *Proc. of ICSLP*, pages 647–650, 1992.

[64] E.Jackson, D.Appelt, J.Bear, R.Moore, and A.Podlozny. A template matcher for robust NL interpretation. In *Proc. of DARPA Speech & Natural Language Workshop*, pages 190–194, 1991.

[65] P.J.Hayes, A.G.Hauptmann, J.G.Carbonell, and M.Tomita. Parsing spoken language: a semantic caseframe approach. In *Proc. of COLING*, pages 587–592, 1986.

[66] W.Ward. Understanding spontaneous speech: The PHOENIX system. In *Proc. of IEEE-ICASSP*, pages 365–367, 1991.

[67] E.Charniak. A neat theory of marker passing. In *Proc. of AAAI*, pages 584–588, 1986.

[68] D.L.Waltz and J.B.Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. In *Cognitive Science*, volume 9, pages 51–74, 1985.

[69] R.Pieraccini, E.Tzoukermann, Z.Gorelov, J-L.Gauvain, E.Levin, C-H Lee, and J.G.Wilpon. A speech understanding system based on statistical representation of semantics. In *Proc. of IEEE-ICASSP*, volume 1, pages 193–196, 1992.

[70] J.Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan and Kaufman, 1987.

[71] H.Tsuboi and Y.Takebayashi. A real-time task-oriented speech understanding system using keyword-spotting. In *Proc. of IEEE-ICASSP*, volume 1, pages 197–200, 1992.

[72] J.R.Rohlicek, W.Russell, S.Roukos, and H.Gish. Continuous hidden Markov modeling for speaker independent word spotting. In *Proc. of IEEE-ICASSP*, pages 627–630, 1989.

[73] J.G.Wilpon, L.R.Rabiner, C.H.Lee, and E.R.Goldman. Automatic recognition of keywords in unconstrained speech using hidden Markov models. *IEEE Trans. Acoust., Speech & Signal Process.*, 38(11):1870–1878, 1990.

[74] R.C.Rose and D.B.Paul. A hidden Markov model based keyword recognition system. In *Proc. of IEEE-ICASSP*, pages 129–132, 1990.

[75] M.Weintraub. Keyword-spotting using SRI's DECIPHER large-vocabulary speech-recognition system. In *Proc. of IEEE-ICASSP*, volume 2, pages 463–466, 1993.

[76] C.E.Shannon. Prediction and entropy of printed English. *Bell Syst. Tech. J.*, 30(1):50–64, 1951.

[77] J.R.Rohlicek, P.Jeanrenaud, K.Ng, H.Gish, B.Musicus, and M.Siu. Phonetic training and language modeling for word spotting. In *Proc. of IEEE-ICASSP*, volume 2, pages 459–462, 1993.

[78] P.Baggia and C.Rullent. Partial parsing as a robust parsing strategy. In *Proc. of IEEE-ICASSP*, volume 2, pages 123–126, 1993.

[79] S.Seneff. Robust parsing for spoken language systems. In *Proc. of IEEE-ICASSP*, volume 1, pages 189–192, 1992.

[80] D.Stallard and R.Bobrow. Fragment processing in the DELPHI system. In *Proc. of DARPA Speech & Natural Language Workshop*, pages 305–310, 1992.

List of Publications by the Author

Major Publications

[1] S.Doshita, T.Kawahara, Y.Mizutani, H.Kojima, M.Ishikawa, and S.Kitazawa. Speaker-independent discrimination of Japanese consonants in isolated syllables using pair-wise discrimination method (in Japanese). *J. Acoust. Soc. Japan*, 45(11):827–836, 1989.

[2] T.Kawahara, S.Doshita, and S.Kitazawa. Speaker-independent consonant recognition by integrating discriminant analysis and HMM (in Japanese; translated in *Systems and Computers in Japan*). *Trans. IEICE*, J73-DII(9):1363–1372, 1990.

[3] T.Kawahara and S.Doshita. Speech recognition by continuous HMM based on pair-wise Bayes classifiers (in Japanese). *Trans. IEICE*, J75-DII(10):1641–1648, 1992.

[4] T.Kawahara and S.Doshita. Comparison of discrete and continuous classifier-based HMM. *J. of Acoust. Soc. Japan (E)*, 13(6):361–367, 1992.

[5] T.Kawahara, S.Matsumoto, and S.Doshita. Continuous speech recognition based on A* search with word-pair constraint as heuristics (in Japanese; translated in *Systems and Computers in Japan*). *Trans. IEICE*, J77-DII(1):1–8, 1994.

[6] T.Kawahara, T.Munetsugu, and S.Doshita. Word spotting in spontaneous speech with heuristic language model (in Japanese). *Trans. IEICE*, (to appear).

[7] T.Kawahara, T.Ogawa, S.Kitazawa, and S.Doshita. Phoneme recognition by combining Bayesian linear discriminations of selected pairs of classes. In *Proc. Int'l Conf. on Spoken Language Processing*, 7.8, 1990.

[8] T.Kawahara and S.Doshita. Phoneme recognition by combining discriminant analysis

and HMM. In *Proc. IEEE Int'l Conf. Acoust., Speech & Signal Process.*, pages 557–560, 1991.

[9] P.Fung, T.Kawahara, and S.Doshita. Unsupervised speaker normalization by speaker Markov model converter for speaker-independent speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, pages 1111–1115, 1991.

[10] T.Kawahara and S.Doshita. HMM based on pair-wise Bayes classifiers. In *Proc. IEEE Int'l Conf. Acoust., Speech & Signal Process.*, volume 1, pages 365–368, 1992.

[11] T.Kawahara, S.Matsumoto, and S.Doshita. A*-admissible context-free parsing on HMM trellis for speech understanding. In *Proc. Pacific Rim Int'l Conf. on Artificial Intelligence*, volume 2, pages 1203–1208, 1992.

[12] T.Kawahara, M.Araki, and S.Doshita. Reducing syntactic perplexity of user utterances with automaton dialogue model. In *Int'l Sympo. on Spoken Dialogue*, pages 65–68, 1993.

[13] T.Kawahara, M.Araki, and S.Doshita. Heuristic search integrating syntactic, semantic and dialog-level constraints. In *Proc. IEEE Int'l Conf. Acoust., Speech & Signal Process.*, volume 2, pages 25–28, 1994.

[14] T.Kawahara, T.Munetsugu, N.Kitaoka, and S.Doshita. Keyword and phrase spotting with heuristic language model. In *Proc. Int'l Conf. on Spoken Language Processing*, volume 2, pages 815–818, 1994.

Technical Reports

[1] T.Kawahara, Y.Mizutani, S.Kitazawa, and S.Doshita. Application of pair-wise discrimination method to Japanese consonant recognition. *STUDIA PHONOLOGICA*, 22:83–93, 1988.

[2] T.Kawahara and S.Doshita. Speaker-independent consonant recognition by integrating discriminant analysis and Markov modeling (in Japanese). In *IEICE Tech. Report*, SP89-26, 1989.

[3] T.Kawahara, S.Doshita, and S.Kitazawa. Speaker-independent consonant recognition by integrating discriminant analysis and HMM. *STUDIA PHONOLOGICA*, 23:33–43, 1989.

[4] S.Matsumoto, T.Kawahara, and S.Doshita. Spoken language understanding based on A* search integrating lexical, syntactic and semantic constraints (in Japanese). In *IEICE Tech. Report*, SP91-93, NLC91-50, 1991.

[5] P.Fung, T.Kawahara, S.Doshita, and M.Adda. Improved speaker Markov modelling for unsupervised speaker normalization. *STUDIA PHONOLOGICA*, 25:49–58, 1991.

[6] T.Kawahara and S.Doshita. Evaluation of continuous and discrete HMM based on pair-wise Bayes classifiers (in Japanese). In *IEICE Tech. Report*, SP92-21, 1992.

[7] T.Kawahara. Search algorithms for speech recognition – introducing A* search – (in Japanese). In *IEICE Tech. Report*, SP92-36, 1992.

[8] T.Kawahara, S.Matsumoto, N.Nukaga, and S.Doshita. Evaluation of A* search using word-pair constraint as heuristics (in Japanese). In *IEICE Tech. Report*, SP92-100, 1992.

[9] T.Munetsugu, T.Kawahara, M.Araki, and S.Doshita. Keyword spotting for spontaneous speech understanding (in Japanese). In *IEICE Tech. Report*, SP92-116, 1993.

[10] N.Kitaoka, T.Kawahara, and S.Doshita. Continuous speech recognition based on right-to-left A* search with case structure (in Japanese). In *IEICE Tech. Report*, SP93-19, 1993.

[11] N.Kitaoka, T.Kawahara, and S.Doshita. Phrase spotting for spontaneous speech understanding (in Japanese). In *IEICE Tech. Report*, SP93-116, NLC93-56, 1993.

[12] M.Nukaga, T.Kawahara, and S.Doshita. A Kansei processing model of voice quality evaluation (in Japanese). In *IEICE Tech. Report*, HC93-67, 1994.

[13] T.Kawahara, M.Araki, and S.Doshita. Development of spoken language understanding system – comparison of syntax-driven and keyword-driven approach –. *STUDIA PHONOLOGICA*, 27:43–54, 1993.

[14] R.Satoh, T.Kawahara, and S.Doshita. Unknown word handling in speech recognition system based on syntax-driven A* search (in Japanese). In *IEICE Tech. Report*, SP94-25, 1994.

[15] T.Kawahara, T.Munetsugu, K.Miki, and S.Doshita. Comparison of language models for keyword spotting in spontaneous speech (in Japanese). In *IEICE Tech. Report*, SP94-28, 1994.

[16] T.Kawahara, N.Kitaoka, and S.Doshita. Robust spoken language understanding based on phrase spotting (in Japanese). In *IEICE Tech. Report*, SP94-68, NLC94-37 (94-SLP-4-6), 1994.

Oral Presentations

[1] T.Kawahara and S.Doshita. Speaker-independent recognition of Japanese consonants using discriminant analysis and Markov modeling (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-P-13, spring 1989.

[2] Y.Mizutani, T.Kawahara, and S.Doshita. Recognition of continuous vowels and semi-vowels based on the transition rules in the phoneme sequences obtained by discriminant analysis (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-1-13, autumn 1989.

[3] T.Kawahara and S.Doshita. A study on HMM with symbol prediction (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-3-11, spring 1990.

[4] T.Ogawa, T.Kawahara, and S.Doshita. An algorithm to select effective pairs in pair-wise discrimination method (in Japanese). In *Proc. Annual Convention IPSJ*, 7M-1, autumn 1990.

[5] T.Kawahara, H.Maki, and S.Doshita. Neural network based on piece-wise linear discriminant functions (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-P-16, autumn 1990.

[6] T.Kawahara. HMM based on Bayes classifier (in Japanese). In *Symposium on HMM and NN*, SPREC90-2, pages 33–36, 1991.

[7] P.Fung, T.Kawahara, and S.Doshita. Unsupervised speaker normalization by speaker Markov model converter for speaker-independent speech recognition systems. In *Proc. Annual Convention IPSJ*, 6D-1, spring 1991.

[8] T.Kawahara and S.Doshita. Continuous HMM based on pair-wise classification (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 3-P-17, spring 1991.

[9] T.Kawahara and S.Doshita. Evaluation of pair-wise Bayes classifier based HMM (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 3-5-12, autumn 1991.

[10] S.Matsumoto, T.Kawahara, and S.Doshita. Spoken language understanding based on A* search using lexical constraints as heuristics (in Japanese). In *Proc. Annual Convention IPSJ*, 6V-4, autumn 1991.

[11] T.Kawahara and M.Araki. Comparison of left-to-right A* search and keyword-driven search (in Japanese). In *Symposium on Continuous Speech Recognition*, SPREC91-2, pages 29–32, 1992.

[12] T.Kawahara and S.Doshita. Comparison of discrete and continuous classifier-based HMM (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-1-6, spring 1992.

[13] T.Kawahara, S.Matsumoto, and S.Doshita. Incorporating probabilistic context free grammar and semantic constraint into A* search (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-1-7, autumn 1992.

[14] N.Kitaoka, T.Kawahara, and S.Doshita. Phoneme recognition for both sexes using pair-wise discrimination method with multiple templates (in Japanese). In *Proc. Annual Convention IPSJ*, volume 2, pages 1–2, autumn 1992.

[15] T.Kawahara, M.Araki, T.Munetsugu, and S.Doshita. Comparison of left-to-right parser and keyword-driven parser for spontaneous speech understanding (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 3-4-9, spring 1993.

[16] T.Kawahara and M.Araki. A new computational model for spontaneous speech understanding (in Japanese). In *Symposium on Spontaneous Speech Processing*, SPREC93-1, pages 49–53, 1993.

[17] N.Kitaoka, T.Kawahara, and S.Doshita. Continuous speech recognition using right-to-left parser (in Japanese). In *Proc. Annual Conf. JSAI*, pages 507–510, 1993.

[18] T.Kawahara, M.Araki, K.Kohda, N.Nukaga, and S.Doshita. On the use of dialogue-level knowledge to improve speech recognition (in Japanese). In *Proc. IEICE Conf.*, SA-6-3, autumn 1993.

[19] T.Kawahara, M.Araki, and S.Doshita. Syntactic prediction of next user utterances with dialogue state transition model (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-8-12, autumn 1993.

[20] M.Nukaga, K.Iino, T.Kawahara, and S.Doshita. A Kansei processing model of voice quality evaluation (in Japanese). In *Proc. Annual Convention IPSJ*, volume 2, pages 263–264, autumn 1993.

[21] T.Kawahara, N.Kitaoka, and S.Doshita. Speech understanding based on phrase spotting (in Japanese). In *Symposium on Speech Recognition and Understanding*, SPREC93-3, pages 33–35, 1994.

[22] T.Kawahara and S.Doshita. On the heuristics for A* search in continuous speech recognition (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-7-6, spring 1994.

[23] T.Kawahara, T.Munetsugu, and S.Doshita. Word spotting method using lexical and syllable knowledge as heuristics (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-7-7, spring 1994.

[24] T.Kawahara, N.Kitaoka, and S.Doshita. Spontaneous speech recognition based on A*-admissible phrase spotting (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-8-15, autumn 1994.

[25] K.Miki, T.Kawahara, and S.Doshita. Rejection of utterances out of syntax of the task (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-Q-1, autumn 1994.

Appendix

I: List of 653 Words used for Phone Model Evaluation

pari	piza	puro	peN	poozu
papa	papirusu	napukiN	kapera	aporo
waipaa	taipiN	paipu	taipei	paipo
rupaN	karupisu	supuN	boorupeN	rupo
depaato	sepia	epuroN	hudepeN	repooto
popai	kopii	popura	opera	toporozii
taNpa	saNpi	oNpu	moNpe	siNpo
pya		pyuuma		pyootoru
ipya		rapyuta		ipyo
upya		epyu		opyo
boNpyaku		koNpyuuta		kaNpyo

tai	tii		te	to
kata	ratisu		tate	ato
ita	sitii		mite	ito
uta	butikku		uteN	utoi
geta	betii		ete	eto
botaN	kotii		kote	oto
iNtai	aNtiiku		aNtei	seNto

kazi	kiri	kuma	ke	koi
aka	aki	aku	take	kako
ika	iki	iku	ike	ikoi
ukai	uki	uku	ukeru	ukoN
ekaki	eki	ekubo	sekeN	neko
oka	oki	oku	oke	soko
saNka	kiNki	moNku	kaNkei	haNko
kyarameru		kyuusyuu		kyouto
kaikyaku		tsuikyuu		dakyou
okyaku		mukyuu		ekyou
kaNkyaku		reNkyu		kaNkyo

basu	biN	buka	beso	boki
kaba	tabi	abu	kabe	saboru
ibara	ibiki	ibu	kibeN	ibo
uba	rubii	ubu	sube	zuboN
deba	ebi	ebumi	meberi	ebosi
oba	obi	kobu	nobe	oboN
aNba	toNbi	kaNbu	huNbetsu	toNbo
byakuya		byuuroo		byou
kaibyaku		iNtabyuu		ibyo
obya		rebyuu		hatsubyou
saNbyaku		saNbyuu		kaNbyou
damu	disuku		demo	doki
ada	dadii		tade	kado
idai	kidii		ideN	ido
udaru	audii		ude	udo
eda	edi		edeN	edo
odate	bodii		odeN	odori
haNda	haNdii		hoNdeN	kaNdo
gaka	gisi	guN	geki	gomi
taga	hagi	kagu	age	ago
iga	igi	kigu	hige	igo
sugata	mugi	hugu	kuge	tsugou
kega	negi	eguru	kegeN	ego
ogamu	nogiku	kogu	toge	ogori
iNga	eNgi	teNgu	maNgetsu	koNgo
gyaku		gyuuniku		gyouzi
kagyaku		nikugyuu		igyou
zigyaku		regyuraa		gyogyou
haNgyaku		keNgyuu		niNgyou
raN	rika	rusu	rei	roze
sara	ari	aru	kare	aroe
irai	kiri	iru	nire	iro
ura	uri	uru	mure	huro
era	eri	eru	tere	tero
sora	ori	oru	kore	doro
koNraN	kaNri	siNrui	seNrei	seNro
ryaku		ryuuki		ryokou
sakuryaku		karyuu		iryou
geryaku		kiryuu		yoryoku
seNryaku		daNryuu		siNryou
machi	mizu	musi	me	mori

ama	ami	amu	ame	kamo
ima	imi	imu	hime	imo
uma	umi	umu	ume	kumo
ema	emi	nemuru	seme	memo
goma	tomi	nomu	kome	momo
saNma	chiNmi	kiNmu	moNme	hiNmoku
myaku		myuNheN		myou
kimyaku		kamyu		kimyou
doumyaku		emyureeto		zyumyou
buNmyaku		saNmyuuzikku		siNmyou

nasi	nizi	numa	netsu	nori
ana	ani	kanuu	ane	kano
ina	hinichi	inu	ine	mino
huna	uni	kunugi	une	nuno
senaka	zeni	tenuki	keneN	enoki
kona	oni	konu	one	ono
oNna	kiNniku	seNnuki	kaNneN	teNnou
nya		nyuuka		nyozitsu
inya		kanyuu		tounyou
rounyaku		inyuu		senyooru
haNnya		haNnyuu		fuNnyou

hako	hima	husi	heta	hosi
haha	mahi	ahure	aheN	aho
ihai	ihiN	ihu	iheN	mihoN
uha	buhiN	ruhu	uheN	muhoN
nehaN	zehi	ehu	teheN	ehoN
ohagi	gohiki	sohu	hohei	toho
zeNhaN	neNhitsu	siNhuzeN	siNheiki	kiNhoNi
hyaku		hyuuga		hyouka
nihyaku		nakahyuuga		zuhyou
gohyaku		kamihyuuga		sehyou
seNhyaku		bizeNhyutte		saNhyou

faamu	fiiriNgu		feezu	foomu
rifarensu	pasifikku		kafe	pafoomansu
arufaa	ofisu		efekuto	kariforunia
saNfaN	saNfiito		aNfea	siNfonii

saka	sio	suna	semi	soto
asa	asi	asu	ase	aso
isamu	isi	isu	ise	iso
usagi	usi	usu	useru	uso
esa	desi	mesu	zesei	heso

osa	tosi	osu	oseN	osoi	
siNsa	kiNsi	taNsu	meNsetsu	oNso	
syazai		syuwa	syea	syori	
musya		yasyu	bodiisyeepu	tsumesyo	
esyaku		bisyu	kakusyerutaa	gosyo	
kaNsya		geNsyu	saNsyeido	gaNsyo	

zaru	ziko	zuga	zemi	zou	
aza	azi	kazu	aze	nazo	
hiza	izi	izu	izeN	mizo	
muzaN	uzi	uzu	huzei	buzoku	
gezaN	eziki	ezu	zeze	ezo	
goza	ozi	mozu	oze	hozo	
neNza	haNzi	aNzu	meNzei	maNzoku	
zyama		zyuu	zyettoki	zyosi	
suizyaku		yazyuu	negurizye	gezyo	
hakuzyaku		gezyuN	aruzyeria	zyozyou	
siNzya		seNzyutsu	eNzyeru	geNzyou	

chairo	chiri	chuuwa	cheeN	chouwa	
akacha		kaichuu	karuche	syachou	
mecha		zyochuu	imecheN	gichou	
baNcha	oNchi	kaNchuu	saNcheeN	keNcho	
hachi	ichi	uchi	echigo	bochi	
tsume					
atsui	itsu	utsu	tetsu	otsu	paNtsu

atama	isiki	udoN	esi	ochiba	

wa					
kawa	iwa	suwa	kewasii	owari	kaNwa

ya		yu		yo	
kaya		ayu		tayori	
iya		chiyu		iyo	
tsuya		tsuyu		uyoku	
heya		neyuki		seyo	
oya		oyu		oyogu	
aNyaku		kaNyu		kaNyo	

naate	ai	au	ae	ao	aN
iatsu	suii	siuchi	ie	siori	iN
suasi	ui	suu	ue	uo	uN
teate	ei	keu	maee	neoki	eN
doa	oi	ou	oe	anooka	oN

`daNatsu`	`keNi`	`iNutsu`	`uNei`	`keNo`
`maazi`	`siizuN`	`suupu`	`peesu`	`tooki`

II: List of 50 Sample Grammatical Sentences

Note: English translation is for reference.

1. `asu no niji kara saNji made nikeN de oNsee keNkyuukai o hiraki tai`
   (I want to hold a speech group meeting at room No. 2 from 2 to 3 o'clock tomorrow.)
2. `hai`
   (O.K.)
3. `juugatsu saNjuunichi ni jiNkoochinoogakai no geNkoo no shimekiri ga arimasu`
   (Oct. 30th is the deadline of the paper for Society of Artificial Intelligence.)
4. `kekoo desu`
   (No, thank you.)
5. `hatsuka kara nijuusaNnichi made joohooshorigakai de nagoya ni shuchoo shimasu`
   (I will make a business trip to Nagoya from the 20th to the 23rd for a conference of Information Processing Society.)
6. `asate no gozeN juuji kara juuniji made seminaashitsu de uchiawase o okonau`
   (I will have a meeting at the seminar room from 10 to 12 A.M. on the day after tomorrow.)
7. `iie juuniji made desu`
   (No, until 12 o'clock.)
8. `soodesu`
   (That's right.)
9. `asu no kuji kara kaigi o okonau`
   (I will have a meeting at 9 o'clock tomorrow.)
10. `juuichiji made desu`
    (Until 11 o'clock.)

11. `daikaigishitsu desu`
    (At the large meeting room.)

12. `juunananichi ni kyuuka o tori tai`
    (I want to take a holiday on 17th.)

13. `jaa yame masu`
    (Then, I'll cancel the request.)

14. `nijuugonichi no juusaNji kara juurokuji made ichikeN de yosaNkaigi o hiraki tai`
    (I want to have a budget meeting at room No. 1 from 13 to 16 o'clock on the 25th.)

15. `juugoji made nara daijoobudesuka`
    (Can you attend if it is until 15 o'clock.)

16. `raishuu no kayoobi no juuyoji kara ichiji kaN saNkoo de teeseesuiroN keNkyuukai o okonai tai`
    (I want to have a qualitative reasoning group meeting for an hour from 14 o'clock next Tuesday.)

17. `juurokuji ikoo no sukejuuru wa doonateimasuka`
    (What is the schedule after 16 o'clock ?)

18. `dewa juurokuji kara ni shimasu`
    (Then, I will start it at 16 o'clock.)

19. `juusaNnichi wa haNdai deno shizeNgeNgoshori keNkyuukai ni shuseki suru`
    (On the 13th, I will attend the natural language group meeting at Osaka Univ.)

20. `juuji ikoo nara aiteimasuka`
    (Are you free after 10 o'clock ?)

21. `dewa juuji kara dekake masu`
    (Then, I will go out at 10 o'clock.)

22. `asu no buNkakai o gogo saNji kara ni heNkoo suru`
    (Change the time of tomorrow's sub-committee meeting to 3 P.M.)

23. `juuhachinichi no kaigi no basho o dainieNshuushitsu ni heNkoo suru`
    (Change the place of the meeting on the 18th to seminar room No. 2.)

24. `iie dainieNshuushitsu desu`
    (No, at seminar room No. 2.)

25. `asu no oNsee keNkyuukai o chuushi suru`
    (Cancel tomorrow's speech group meeting.)

26. `aa asate deshita`
    (Oh, it is the day after tomorrow.)

27. `juugonichi no tokubetsukooeN wa chuushi ni narimashita`
    (The special lecture on the 15th is canceled.)

28. `tokubetsukooeN wa itsu desuka`
    (When will the special lecture be held ?)

29. `sore ga chuushi ni narimashita`
    (It is canceled.)

30. `asate no kaigi o juurokuji kara ni heNkoo suru`
    (Change the time of the meeting on the day after tomorrow to 16 o'clock.)

31. `dewa juugoji kara ni shimasu`
    (Then, change it to 15 o'clock.)

32. `kiNyoobi no yosaNkaigi no basho o shookaigishitsu ni heNkoo shitai`
    (I want to change the place of the budget meeting next Friday to the small meeting room.)

33. `dewa sonomama de iidesu`
    (Then, leave it as it is.)

34. `nijuusaNnichi no oNsee keNkyuukai o kuji kara ni heNkoo shitai`
    (Change the time of the speech group meeting on the 23rd to 9 o'clock.)

35. `juuichiji ikoo no yotee wa naniga arimasuka`
    (What are the plans after 11 o'clock ?)

36. `dewa juusaNji kara juugoji made ni heNkoo shimasu`
    (Then, make it 13 to 15 o'clock.)

37. `kyoo no seminaa wa naNji kara desuka`
    (What time does today's seminar start ?)

38. `juugonichi no beNkyookai wa naNji kara desuka`
    (What time will the seminar on the 15th start ?)

39. `naNji made desuka`
    (What time will it end ?)

40. `pari shuchoo wa naNnichi kara naNnichi made desuka`
    (On which days is your business trip to Paris scheduled ?)

41. `shizeNgeNgoshori keNkyuukai no geNkoo no shimekiri wa itsu desuka`
    (What day is the deadline of the paper for the natural language group ?)

42. `asate no kaigi wa dokode arimasuka`
    (Where will the meeting on the day after tomorrow be held ?)

43. `kyoo no yotee wa naniga arimasuka`
    (What are today's plans ?)

44. `juurokunichi no gogo no yotee wa doonateimasuka`
    (What are the plans for the afternoon of the 16th ?)

45. `asu no keNkyuukai wa dokode`
    (Where will the group meeting be held tomorrow ?)

46. `juurokunichi no hapyookai wa naNji kara`
    (What time will the presentation on the 16th start ?)

47. `aa juunananichi deshita`
    (Oh, it is on the 17th.)

48. `itsuka no teeseesuiroN keNkyuukai wa naNji kara`
    (What time will the qualitative reasoning group meeting on the 5th start ?)

49. `jaa kaNchigai deshita`
    (Then, I was wrong.)

50. `kyoo no gogo wa aiteimasuka`
    (Are you free this afternoon ?)

III: List of 25 Sample Ill-formed Sentences

Note: {..} represents a hesitation. English translation is well-formed.

1. `{etto} raishuu no {.. etto} kayoobi kara {..} mokuyoobi made {..} nagoya ni shuchoo shimasu`
   (I will make a business trip to Nagoya from Tuesday to Thursday next week.)

2. `{ee..} koNshuu no kiNyoobi ni kyuuka o tori taiNdesuga`
   (I would like to take a holiday this Friday.)

3. `jaa {..} yame masu`
   (Then, I'll cancel the request.)

4. `kayoobi no {..} gogo yoji kara {...} kougi o {..} ire tai`
   (I have a lecture at 4 P.M. on Tuesday.)

5. `{eeto ..} raishuu no getsuyoobi {.. ee} gozeN hachiji kara {... ee} seminaa o {..} tooroku shitekudasai`
   (Please input the seminar at 8 A.M. next Monday.)

6. `kiNyoobi {..} no gogo yoji kara {..} tenisu o shimasu`
   (I will play tennis from 4 P.M. next Friday.)

7. `konoegurauNdo de {...} onegaishimasu`
   (At Konoe ground, please.)

8. `{e} getsuyoobi no beNkyookai o {..} gogo niji kara ni {..} heNkoo shitekudasai`
   (Please change the time of the next Monday's seminar to 2 P.M.)

9. `raishuu no {...} shuchoo o {..} torikeshimasu`
   (Cancel the business trip next week.)

10. `raishuu no kayoobi no {..} tenisu {..} no jikaN o heNkoo shimasu`
    (Change the time for playing tennis next Tuesday.)

11. `{e ..} ichiji kara ni heNkoo shitekudasai`
    (Please change it to 1 o'clock.)

12. `{anoo ..} asu no kaigi no basho o heNkoo shimasu`
    (Change the place of tomorrow's meeting.)

13. `{ee} daikaigishitsu de okonai masu`
    (At the large meeting room.)

14. `{anoo ..} saNjuunichi no kaigi wa dokode`
    (Where will the meeting on the 30th be held ?)

15. `{e ..} tenisu no yotee wa itsu deshitaka`
    (On which day is tennis scheduled ?)

16. `{to ..} sono tenisu {..} no yotee o chuushi shitekudasai`
    (Please cancel the tennis.)

17. `{ee} getsuyoobi no {..} kougi o {..} kyuukoo ni shimasu`
    (Cancel the next Monday's lecture.)

18. `kiNyoobi niwa naniga {..} yotee ga haite imashitaka`
    (What are the plans for next Friday ?)

19. `asate no seminaa {..} goji kara deshitaka`
    (Will the seminar on the day after tomorrow start at 5 o'clock ?)

20. `{ee} naNji kara nara aiteimasuka`
    (What time are you free ?)

21. `juuji {..} kara desu`
    (After 10 o'clock.)

22. `{ee ...} naNji made desuka`
    (Until what time ?)

23. `yoka no kougi o {ee} chuushi shimasu`
    (Cancel the lecture on 4th.)

24. `{e} raishuu no kayoobi no tenisu o {..  ee} kaigi ni heNkoo shimasu`
    (Cancel next Tuesday's tennis, and have a meeting instead.)

25. `gozeN kuji kara` {..} `juuniji made`
    (From 9 to 12 in the morning.)